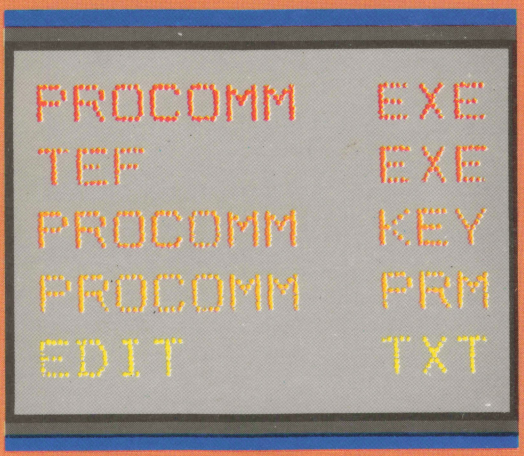
STEP BY STEP

Using MS-DOS on the Amstrad PC



Ian Sinclair

HEINEMANN
NEW TECH

STEP BY STEP

Using MS-DOS on the Amstrad PC

Using MS-DOS on the Amstrad PC

Ian Sinclair

Heinemann London

Heinemann Professional Publishing Ltd
22 Bedford Square, London WC1B 3HH

LONDON MELBOURNE JOHANNESBURG AUCKLAND

First published by Newtech Books Ltd 1987
First published by Heinemann Professional Publishing Ltd 1987

British Library Cataloguing in Publication Data
Sinclair, Ian R.
Using MS-DOS on the Amstrad PC
– (Step by Step)
1. Amstrad PC1512 (Computer) – Programming
2. MS-DOS (Computer operating system)
I Title II Series
005.4'469 QA76.8.A4

ISBN 0 434 91842 3

Designed by John Clark and Associates, Ringwood, Hampshire

Printed in the UK by HGA Printing Co. Ltd, Brentford, Middlesex

Contents

Contents

Contents

Contents

Preface

The PC1512 is the culmination of the Amstrad computers to date – a low-cost machine of enormous capability that will bring to all users the huge range of software designed for this class of machine. We have seen what the impact of the PCW machines was on CP/M software, and we can now expect the same price decreases in software to apply to machines using the MS-DOS type of operating system, for the benefit of all. Once again, this has been done by using straightforward methods, resulting in a low-cost, reliable machine.

Though the PC1512 features the GEM system as a way of making some types of programs easier to use for the inexperienced owner, much more is open to you if you understand the Disk Operating System (DOS) itself, and can dispense with GEM. Not all programs are best served by using GEM, and GEM requires a very substantial chunk of the memory of your machine in order to operate. Do without it, and you have more direct control over the machine, allowing you to get more useful computing done. You can also make much more effective use of the RAM-disk feature, and you may very well find that many actions are simpler to carry out.

This book is aimed at the new PC1512 owner who wants to know how to make the machine work closer to its limits. I don't assume that you can write programs, and I don't intend to teach you. What I do assume is that you want to make good use of the machine, and need to know about the commands that are available. These commands are the commands of the Disk Operating System, referred to from now on as DOS. This is the program collection that controls the machine, particularly as regards inputs and outputs, allowing useful programs to be run. Your PC1512 comes with a choice of two such DOS systems, MS-DOS and DOS Plus, with GEM used as a way of controlling either type of DOS. Of these, the main bulk of software is written for MS-DOS, and you can run any programs that are on compatible systems like MS-DOS and IBM PC-DOS, and which are on suitable disks.

You can run them, that is, if you know how – and that's why this book exists. Computer manuals have to be reference books, full of information that is useful if you know what it refers to. No computer manual can set out to explain in detail – simply because there is always too much to explain. In this book, I have concentrated on a step-by-step approach to the essentials of PC1512 MS-DOS – which means using it to run business software and to manage the disks that you will use. Obviously, experience helps in this respect, and if you

Preface

have used the CP/M operating system of the PCW range of machines, you will find it easier to adapt to PC1512 DOS. If this is your first computer, though, don't despair, because you will find clear explanations here. You should, however, leave Part 8 until you feel that you have gained a thorough practical grasp of the use of the other Parts of this book.

As always, a book such as this has resulted from a lot of work by several people. I am most grateful to Mike Fluskey, Managing Director of Newtech Books, for commissioning this book, and making a machine available, and for the unremitting efforts of all the staff of Newtech, particularly Carol and Robin.

NOTE: Commands that you type and the names of programs are printed in upper case (capitals), though you can type them on your keyboard in either lower case or upper case as you please. The large key that is marked with the reversed-L arrow will be referred to as the RETURN key, though on-screen messages refer to it as ENTER.

IAN SINCLAIR
January 1987

PART ONE

Fundamentals

SECTION 1
Processor and memory

A computer consists of a processor and a memory. Modern computers, like the Amstrad PC1512, contain memory that is almost completely empty. The computer becomes useful when you fill some of the memory with a program, a set of instructions that will make the computer carry out some useful tasks.

These programs are stored on magnetic disks, so that the first thing a computer needs to be able to do is to transfer the data stored on the disks into its memory. This is one of the tasks that is handled by the disk operating system, the **DOS** for short.

In addition, the computer must be able to display data on a screen, and it must be able to receive data from its keyboard, and to a lesser extent, from the mouse. It must be able to send data in suitable form to a paper printer so that you can keep permanent records, and also to store data back on to disks to use over and over again.

These are the minimum requirements, and though some of the toy computers of the past did not possess even these, they are regarded simply as a starting point for today's powerful machines.

The key to all these actions is the DOS.

SECTION 2
DOS and memory

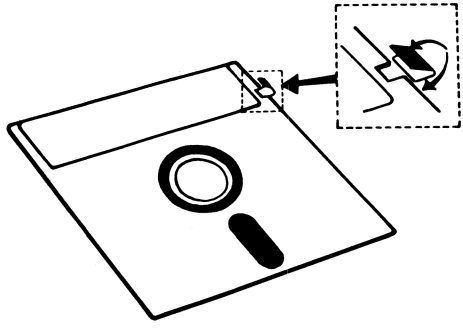
The DOS is another program, but one that is partly held in a part of the memory that is permanent.

Most of the memory of a modern computer is of a type called *volatile memory,* meaning that whatever is stored in the memory disappears when the power supply is shut off. This is the reason for using magnetic disks – when the magnetic powder on the disks is magnetised, the magnetism does *not* disappear after the disk has been recorded.

The computer has a small piece of permanent memory. When you switch on the computer, the instructions in this memory are used to make the computer read in the rest of the DOS from a disk. This disk is called the **System disk,** or **Master disk.**

Without this disk, the computer is useless, so that the first action that you need to carry out when you have your computer set up and ready to run is to make copies of this System disk.

You use one copy for everyday use of the computer, keep another spare in case of trouble, and keep the original System disk, with write-protect tab in place, locked away safely, and, hopefully, never used again.



This makes it sound as if disks are very fragile. They are not, but because the data on the disk is invisible, it's easy to forget that it can be erased.

(Further Information: Amstrad Manual, pages 43 – 44.)

SECTION 3
Care of disks

- Though they are called floppy disks, avoid bending them. This can cause the magnetic coating to flake off.
- Be careful how you insert disks into the drive. It's easy to bend a disk by careless insertion.
- Do not keep disks where they might become wet, as on a windowsill or near a coffee-cup.
- Do not keep disks in very hot places.
- **NEVER** place disks near anything magnetic. This means any device that contains magnets, such as monitors or TV receivers, tape recorders, telephones, loudspeakers or small electric motors.
- Always use backup copies of any valuable disks, and keep the originals in a safe place
- Never touch the disk surface, or keep a disk in a smoky atmosphere.
- Always remove disks from the drive(s) *before* you switch the computer on or off.
- Prepare your labels in advance, and stick one on each disk.
- If you need to write on a disk label when it is in place, use a felt-tip pen, and write lightly to avoid denting the disk.
- Buy the correct disks for the computer. Disks intended for other machines (single-sided or single-density) *often work* with the PC, but are not reliable.

SECTION 4
The DOS in place

Once the DOS has been loaded into the memory, it stays there for as long as the computer is in use, and allows you to control the action of the machine by typing command words. The DOS can use about a tenth of the total memory of the Amstrad machine, and any other programs that you load in will make use of the rest as needed.

You can also make use of the GEM 'front-end', which is a program that allows you to make use of the DOS without using the DOS commands.

GEM makes use of the mouse and the pictures, called *icons*, on the screen, and it takes up a very large amount of memory. In this book, we shall ignore GEM, and concentrate on the use of the DOS. This is because so much more can be accomplished if you understand how to use the DOS. In particular, you can work more quickly and make better use of the memory.

A very few actions, notably on the directory of a hard disk, are more easily dealt with by GEM, and we'll note these as we come to them.

SECTION 5
Master disks

The Master disks of the Amstrad PC1512 contain two disk systems, MS-DOS and DOS Plus. The MS-DOS system is essentially identical to the type of DOS that is used on IBM machines, and this is the DOS that you will need to work with most of the software that runs on the IBM and similar machines.

DOS Plus is intended to be used with GEM, and many of the programs that are specifically intended to be used with the PC1512 will feature GEM and this DOS. In particular, the BASIC 2 programming language that is supplied on your Master disks makes use of GEM and DOS Plus.

The differences between the systems are not, in fact, as great as you might expect, and many programs that are written for MS-DOS will run when you are using DOS Plus. The main point is that GEM is intended to work along with DOS Plus and many MS-DOS programs have not been intended to work with GEM.

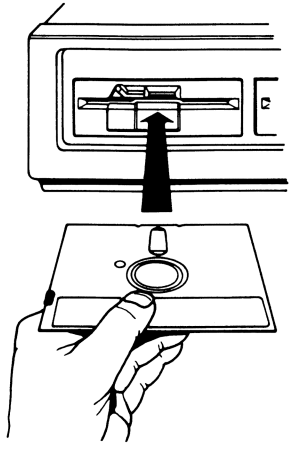
GEM, however, is very demanding of memory, and if you want to use programs that work with a lot of data, then you will probably want to dispense with GEM.

You will also find that the most popular business programs that were written for the IBM machine originally and are now supplied at sensible prices for the PC1512, will not require GEM, and will not be able to use GEM, so that you need to dispense with GEM in order to use them to best advantage.

SECTION 6
What is the disk drive?

To understand the principles of the DOS you need first to understand what the disk drive is and what it does. Unlike tape, which is pulled in a straight line past a recording and replay magnetic head, a disk spins around its centre.

When you insert a disk into a drive, and activate the drive, a hub engages the central hole of the disk, clamps it, and starts to spin it at a speed of about 300 revolutions per minute. The disk itself is a circular flat piece of plastic coated with magnetic material on each side. It is enclosed in a floppy plastic envelope to reduce the chances of damage to the surface.



The hub part of the disk is also built up in plastic to avoid damage to the disk surface when it is gripped by the drive. The surface of each disk is smooth and flat, and any physical damage, such as a fingerprint or a scratch, can cause loss of recorded data. The jacket has slots cut into it so that the disk drive head can touch the disk at the correct places.

There is a slot for the disk head on each side of the disk, and also a hole for locating the disk position. The disk surface can be seen and touched through this slot, and you should *not* touch the disk surface, sneeze on it, or do anything that could leave any marks on the surface.

■ SECTION 7
Disk heads

Through the slot that is cut in the envelope, the heads of the disk drive can touch the surface of the disk, one head on each side. These heads are tiny electromagnets, and each head is used both for writing data and for reading data.

When a head writes data, electrical signals through the coils of wire in the head cause changes of magnetism. These in turn magnetise the disk surface. When the head is used for reading, the changing magnetism of the disk as it turns causes electrical signals to be generated in the coils of wire.

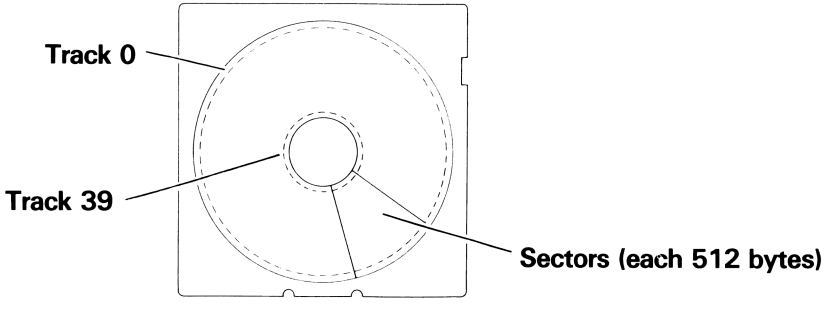
This recording and replaying action is very similar to that of a cassette recorder, with one important difference. Cassette recorders were never designed to record digital signals from computers, but the disk head is.

The reliability of recording on a disk is therefore very much better than could ever be obtained from a cassette, which is why computers for serious use never feature the use of ordinary audio cassettes.

SECTION 8
Head movement

Unlike the head of a cassette recorder, which does not move once it is in contact with the tape, the heads of the disk drive move quite a lot. If the head is held steady, the spinning disk will allow a circular strip (sometimes referred to as a *cylinder)* of the magnetic material to be affected by the head.

By moving the head in and out, to and from the centre of the disk, the drive can make contact with different circular strips of the disk. These strips are called **tracks**. Unlike the groove of a conventional record, these are circular, not spiral, and they are not grooves cut into the disk. The track is invisible, just as the recording on a tape is invisible. What creates the tracks is the movement of the recording/replay head of the disk drive.



A rather similar situation is the choice of twin-track or four-track on cassette tapes. The same tape can be recorded with two or four tracks depending on the heads that are used by the cassette recorder. There is nothing on the tape that guides the heads, or which indicates to you how many tracks exist.

■ SECTION 9
Number of tracks

The number of tracks that you use therefore depends on your disk drives. Like the vast majority of disk drives for other machines, the PC 1512 uses 5¼-inch disks with 40 tracks on each side. These 40-track disks use 48 tracks per inch, so that the useable width of the disk surface is only 40/48 inch, about 0.83 inch.

These disks are known as double-density, double-sided , abbreviated to DSDD, and every disk supplier should have stocks, unlike the 3-inch disks of the earlier Amstrad PCW machines.

In general, if you shop around, particularly among postal suppliers and at exhibitions, you can get packs of ten disks at less than £10.

If you already have disks of a different standard, such as single-sided, single-density, do not throw them away. Try to format them on the PC, and if they format correctly, keep them for non-essential purposes. Do not try to use any disk that does not format correctly.

(Further Information : Amstrad Manual, page 41.)

SECTION 10
Sectors

Once you have accepted the idea of invisible tracks, it's not quite so difficult to accept also that each track can be invisibly divided up. The reason for this is organisation – the data is divided into 'blocks', or **sectors,** each of 512 bytes.

A byte is the unit of computer data; it's the amount of memory that is needed for storing one typed character, for example. Each track of the disk is divided into a number of sectors, and each of these sectors can store 512 bytes of data.

MS-DOS uses 9 sectors per track on a disk, allowing $512 \times 9 = 4608$ bytes to be recorded on each of the 80 tracks (40 on each side). On disks that contain the MS-DOS tracks, 13 tracks are reserved for holding the hidden DOS files, leaving 67 tracks for your use.

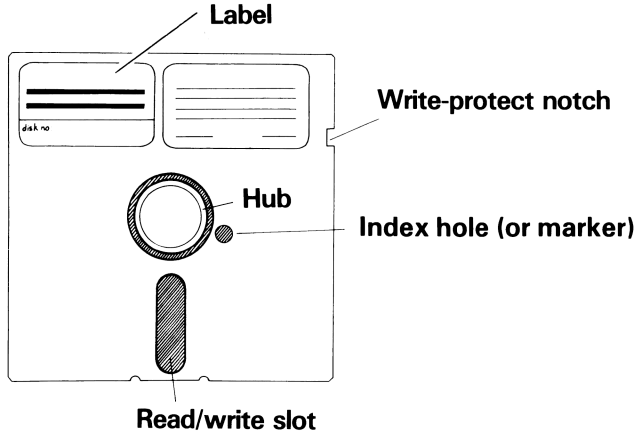
In other tracks, 2 sectors are reserved for the main 'directory' entries, leaving the rest free. This corresponds to a total of 291,840 bytes free on such a disk.

If you format a disk *without* copying over the MS-DOS files, then all of the disk is available, amounting to 362,496 bytes, which is 354K. This type of disk is a data-only disk. Unlike a disk with MS-DOS tracks, you cannot insert a data disk after switching on the machine in order to load the MS-DOS system.

■ SECTION 11
Soft-sectoring

The next thing that we have to consider is how the sectors are marked out. Once again, this is not a visible marking, but a magnetic one.

The system is called **soft-sectoring**. Each disk has a small hole punched into it at a distance of about 25 mm from the centre. There is a hole cut also through the disk jacket, so that when the disk is rotated, it is possible to see right through the hole when it comes round.



When the disk is held in the disk drive, and spun, this position can be detected, using a beam of light. This is the **marker,** and the head can use this as a starting point, putting a signal on to the disk at this position, and at eight others, equally spaced, so as to form sectors.

This sector marking has to be carried out on each track of the disk, as part of the operation that is called **formatting.**

■ SECTION 12
The DOS action

When you load a program from a disk, or save data on a disk, you don't have to worry about the tracks and sectors. You don't, for example, have to specify at which sector and on which track the recording must start, and what to do if there is already something recorded on some of the sectors.

All of this is the action of the DOS, a sort of good-housekeeping system for disks. The action of the DOS is to ensure that a disk is of the correct format, to keep a record of what sectors are used for what files, and to allocate sectors as needed when data is recorded.

The simplest types of DOS, as were supplied in the early days of microcomputers, did little more than this, but were still a very considerable aid to the effective use of computers.

Systems such as CP/M (which first emerged in 1972) added very much more to disk operating systems. The aim of CP/M was to provide an operating system that could be used by a variety of different computers and which would allow software to run on any machine that used the DOS.

This excellent aim was frustrated to some extent by the variety of specifications that grew up for 5¼-inch disks, but the dream was always a reality for the machines that used the 8-inch IBM-type disks.

Now, once more, we are using a standard set by IBM, this time for the 5¼-inch disks. The other point about CP/M was that it provided for much more than just disk operation. It provided a complete system for the control of all inputs and outputs, so that the computer manufacturer could concentrate on designing a machine, knowing that the use of CP/M would allow any of the vast range of software written for the machine to be used.

■ SECTION 13
Sixteen-bit DOS

The arrival of 16-bit computers meant that the older type of CP/M, based on the Z80 microprocessor, had to be supplemented by a newer version.

Two competing systems emerged: **MS-DOS** from Microsoft, and **CP/M-86** from Digital Research. The difference in the licensing prices proved to be the deciding factor, and most 16-bit machines adopted MS-DOS, with the result that most business software for the IBM and similar machines is written so as to run with MS-DOS.

The two systems are, however, so similar that when you can use one you can use the other with only a little more effort. Throughout this book, then, we'll refer to MS-DOS, but note that many commands are used almost unchanged for DOS Plus, which is a development of CP/M-86. In this way, you'll be able to make really effective use of your PC1512 with software that makes use of either type of system.

SECTION 14
Memory use

When you have loaded in MS-DOS from your System Disk, Red Disk No. 1 in the Amstrad set, you have 468,336 bytes free for other programs. This amount of memory is available for you to use as you please. It allows you, for example, to set aside an amount of RAM-disk (see Appendix B) as large as can be employed on the older Amstrad PCW machines.

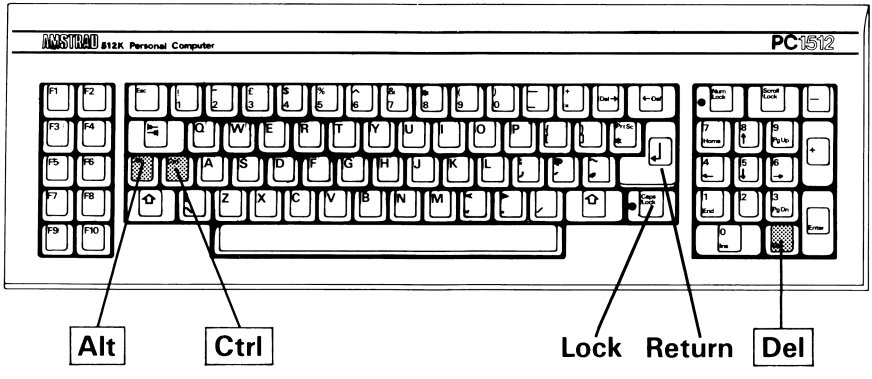
```
TEST

AMSTRAD PC 512k (v1) 22:02 on 06 December 1986
(c)1986 AMSTRAD Consumer Electronics plc

Last used at 05:00 on 06 December 1986

Insert a SYSTEM disk into drive A
Then press any key

Microsoft RAMDrive version 1.16 virtual disk C:
     Disk size: 34k
     Sector size: 128 bytes
     Allocation unit: 1 sectors
     Directory entries: 64

A>ECHO OFF
--- Installing MOUSE   Device Driver V5.00 ---

A>
```

This amount of memory is available for you to use as you please. It allows you, for example, to set aside an amount of RAM-disk (see Appendix B) as large as can be employed on the older Amstrad PCW machines.

■ SECTION 14
Memory use

If you load in GEM, using DOS Plus, the amount of memory you have remaining is so little that you are restricted to 34K of RAM-disk. If you have GEM running, and you want to change to MS-DOS then:

- Remove disk(s) from the drive(s).
- Press the ALT, CTRL and DEL keys together. Note that the DEL key used for this purposes is below the digit 3 on the keypad at the right hand side of the keyboard. The DEL keys on the main part of the keyboard do *not* work in this way.



- When the screen message asks you, insert the MS-DOS System disk (Red disk No. 1), then press any key.

SECTION 15
Files and filenames

Whichever DOS you use will require you to specify names for your files, and both types of DOS have identical requirements. In addition, any programs that make use of the DOS (and practically all do) will impose the same rules, so you need to be familiar with what is required.

To start with, there are names that you cannot and must not use. You cannot use any of the names that appear on the Master Disk, because these are reserved for the programs that bear these names. In addition, there are **internal** commands, stored in the memory, whose names you must not use.

These are:

BREAK CHDIR CD CLS COPY CTTY DATE DEL DIR
ERASE EXIT MKDIR MD PATH PROMPT RENAME
RMDIR RD SET TIME TYPE VER VERIFY VOL

Any attempt to use these names other than for the programs that they represent will cause problems.

Various programs that you use will also have their own prohibitions, all designed to prevent you from damaging essential files on the disks. In general, a suitable filename will consist of up to eight characters, the first of which *must* be a letter. The other characters can be letters, or you can use the digits 0 to 9, or the symbols

$ # & @ ! % ( ) - _ { } ' ~ ^

Unless it's particularly important to you to use these symbols, they are best avoided, particularly symbols like the single inverted quotes, which are easily confused or overlooked. A good rule is to use words with a digit or pair of digits used at the end to express versions, like TEXT1, TEXT2, ..., TEXT15, and so on.

Allowing up to eight characters means that you may have to abbreviate names that you would want to use, like ACCREC1 or BOTLEDG1, but it should be possible to provide a name that conveys to you what the file is all about.

(Further Information: Amstrad Manual, page 93.)

SECTION 16
Drive letter

The eight-character (or less) main name is essential, but there are two other parts to a filename that are optional. One of these is the drive letter.

The PC1512 labels its drives as **A** (the only drive in a single-drive machine), **B** (second drive in a twin-floppy machine) and **C**. The C-drive on a machine that uses only floppy disk(s) will be a RAM-drive. On a machine that uses a hard disk, the C-drive is the hard disk, and the letter **D** will be used for the RAM-drive.

RAM-drive simply means that part of the memory is set aside and used like a disk, so that disk commands will store data to this memory or read data from it. This is useful for storing files that programs often make use of, like the dictionary of a spelling-checker. The standard size of RAM-disk, however, is set at 34K, which is too small for many of the most useful applications. The size of the RAM-drive is changed by the NVR program (See Appendix B).

If you have only a single drive, then you can ignore a lot of the information about drive letters, because only the A drive will be used; and possibly the letter C for RAM-drive.

At the time of writing, however, most of the PC1512 machines were being supplied with at least two floppy drives, and a surprisingly large number were being sold with a hard disk, so that it's most likely that your machine has more than one drive.

By specifying a filename, such as B:PROG, you are stipulating that the file will make use of the disk in drive B. The drive letter precedes the main part of the filename, and is separated from it (very important) with a colon.

If no drive letter is used, of course, the colon is not needed. This use of a drive letter applies whether you want to load the program from the disc or record (save) it on to the disc.

Similarly, using C:PROG will specify the use of the C drive, the RAM-drive or hard disk. Using a drive letter in a filename in this way specifies the drive only for the duration of the command that is used with the filename. In other words, if you have specified that you want to run a program B:TESTIT, then the program is loaded from drive B, but whenever the program has been loaded, the normal drive will be the one that was in use previously.

■ SECTION 16
Drive letter

This normally means that the program loads from drive B, and from then on, any use of disks will make use of the A drive again. If no change is made (using the command B:, for example) then the normal default drive is the A drive. Both MS-DOS and DOS Plus have commands that are particularly aimed at the hard disk user. These are dealt with separately in this book.

(Further Information: Amstrad Manual, page 206.)

■ SECTION 17
Extension

As well as the drive letter, which is always a single letter placed ahead of the filename and separated by a colon, the complete filename can contain an **extension.** The extension is a set of up to three letters that are added to the end of the filename, separated by a dot.

Like the disk drive letter, the extension is optional, and if it is not used, then the dot is not needed. The purpose of the extension is to convey some extra information about the type of file, though you can make whatever use of it suits your own purposes, within limits.

Just as there are forbidden filenames, there are also forbidden extensions, and in particular you should not make use of the extensions .EXE, .BAK, .COM or .OBJ for your files, because these are extensions that are used with special meanings.

The following list shows a few 'standard' extension letters and their uses, and you should, if you use extensions at all, either keep to some of these or use some entirely different codes of your own.

BAK	A backup file.
BAS	A program that cannot be run unless BASIC has been loaded first.
BAT	A batch file of commands to the DOS.
COM	A program that can be loaded and run by typing its (main) name.
DOC	A text file of documentation for a program.
EXE	As for COM, but a longer program.
MSG	A text file of instructions.
SYS	A file that is used by the operating system.
TMP	A file that is created temporarily and wiped later.

Particularly useful extensions for you to use are .TXT for files of text, as from a word processor, and .DAT for files of data as might be used in accounts programs. You will find that many programs that you will use will generate their own extensions for data that they save to disk, and you should not use these extensions for other files.

■ SECTION 17
Extension

It's particularly important to know at this point that any file with the extension of .BAK is a file that has been replaced with a more recent version with the same main filename. Some programs will not allow you to save a file with the same name as one that already exists on the disk. Others will automatically rename the existing file as a BAK file, and save your new file with a different extension. Many programs, however, will delete an old file that has the same name as one you are saving.

(Further Information: Amstrad Manual, page 94.)

SECTION 18
The directory

When you call for a directory of a disk, using the DIR command, then the extensions are shown as part of the filename, but separated by a space rather than a dot. The main name is printed in a space that allows for eight characters, and if it needs fewer then spaces are used to pad it out.

Thus the name that you might type as B:MYFIL.DAT will appear on the directory of drive B as MYFIL DAT, so making it easier to list the directory in neat columns. When you are using floppy disks, you will normally keep a set of files on one disk, and change disks to gain access to another set of files.

```
A>

 Volume in drive A is PCWRITETXT
 Directory of  A:\

WSPREF           1902  17-12-86  15:16
WS1A             6324  17-12-86  14:41
WS1B             3134  17-12-86  17:02
        3 File(s)     349184 bytes free
```

Hard disk users, however, can keep a very large number of files on a disk, and some method of dividing files into groups is needed. This is provided by the directory tree system covered in more detail in Part 4.

■ SECTION 19
Copying the system disks

You are urged to make backup copies of your system disks whenever you start to make use of your PC1512. This is not because disks are unreliable – they are, in fact, quite remarkably reliable – but because the system disks are valuable and could be damaged in continual use.

Disks can be invisibly damaged by magnetism, and if a System disk is damaged in this way, the first that you would know of it would be when something stopped working. You might be lucky – the damage could be in a part of the system that you never use.

On the other hand, you could start to use a common utility like FORMAT and find that it simply locked up the computer so that the keys had no effect. One of the most likely sources of disk **corruption,** as this is called, is switching the machine on or off with a disk engaged in the drive.

Since you make use of the system disks each time you start the machine, these are the disks that are most at risk from this little piece of forgetfulness. The sensible thing to do, then, is to make backup copies of the System disks, and then store the original System disks in a safe place (dry, cool and well away from magnets) until new copies are needed.

The System disks should be protected by placing the write-protect tab on each disk *before* copying. The backup copies are then used for all normal actions that involve the System disks. Most business users will want to make yet another set of copies, and use these for making working copies if a backup is damaged.

■ SECTION 20
Backup

The MS-DOS copy utility program for copying (backing-up) floppy disks is called **DISKCOPY**. It can be used in much the same way whether you have one drive or more. You should, as a precaution, fasten a write-protect tab to each disk that you want to copy.

For a two-drive machine, you should start up the computer in the normal way, so that the prompt A> shows on the screen. You then insert the System disk into drive A, and type the command:

DISKCOPY A: B:

Each command of this kind is executed by pressing the RETURN key. A screen message will then appear.

```
A>
A>diskcopy a: b:

Insert SOURCE diskette in drive A:

Insert TARGET diskette in drive B:

Press any key when ready . . .
```

For two drives, you will be asked to insert the source disk (the System disk in this case) in drive A, and the target disk (the one to which the files will be copied) in drive B. Once you have carried out these actions, you can press any key to continue with the copying. The System disk in drive A will then be read, and after about 40 seconds, another message will appear.

The message will simply be to the effect that the copy is complete, and you will be asked to indicate, by pressing the Y or N keys, whether or not you want to copy the disk again, or to copy another disk. If the target disk was not a formatted disk, then copying takes longer because formatting is carried out first.

If you have a single drive, then the command need only be:

DISKCOPY

and the message will be to insert the source disk in drive A. When the program has read the data from the source disk, you will be asked to place the target disk into drive A and press any key when this has been done.

The process is complete when the 'Copy another diskette (Y/N)' message appears, and if you don't want to copy any more disks, you can leave the program by pressing the N key.

(Further Information: Amstrad Manual page 334.)

■ SECTION 21
The backup disk

Make sure that when you have made a backup that the backup disk is clearly labelled. Remember not to write on any label that has been stuck to the disk envelope – do all of your writing while the label is still attached to the sheet of labels.

If you *must* write on labels that are already attached to disks, then use a special disk-writing pen, such as the one made by Berol. This is arranged so that the point will bend if you use too much pressure. Whatever you do, don't use a ball-point, because this can make marks on the disk itself, making sectors unusable.

Do *not* put a write-protect tab over your backup disks, because the computer often needs to write small amounts of data to these disks. Your Master copies should, however, be protected. If you try to use the protected copies in some actions, you will get a message about write-protection being present, and you will have to change over to an unprotected copy.

SECTION 22
System tracks

If you want to use a blank floppy disk for data only, such as the document files of a word processor, then the normal FORMAT command will be enough.

1 **Place the MS-DOS Master disk in drive A.**

2 **Type FORMAT A: (single drive) or FORMAT B: (twin floppies). Press RETURN.**

3 **Remove the Master disk, and follow the on-screen instructions.**

If, however, you want to use the disk for starting up the machine and loading in one particular program (word processor, spreadsheet, etc.) you can alter the FORMAT command to read FORMAT/S. This will put the MS-DOS system tracks on to the disk so that you can autostart (see Part 7) the program.

You can also place some system tracks on to a disk that has already been formatted and used, by the SYS command:

1 **Place the Master disk in drive A.**

2 **Type SYS B: . If you have two drives, place the formatted disk in drive B, else await instructions to change disks. Press RETURN.**

3 **Follow the on-screen instructions.**

Note, however, that this omits one file called COMMAND.COM, so that the disk will *not* autostart.

(Further Information: Amstrad Manual, pages 335, 337.)

■ SECTION 23
System disk to hard disk copying

The **FDISK** command is used to configure a hard disk, if you have one. The MS-DOS System disk should be in drive A. Continue as follows:

1. **Type FDISK. You will see a menu of options. Select option 1.**
2. **You will be asked if all the disk is to be used (Y/N). Select N.**
3. **You will be shown the size of the disk in 'blocks', and asked 'What size?.'**
4. **Type in a number that is one less than the number of blocks shown. For example, if the message shows 614 blocks, type 613, then press RETURN.**
5. **You are then asked 'From where'. Reply by typing 1, then RETURN.**
6. **The main menu returns. Select option 4 to return to MS-DOS**
7. **Now type FORMAT C: to format your hard disk**
8. **Now type FDISK again. Choose option 1 from the menu again**
9. **When you are asked 'Do you want all of the disk', type Y this time (RETURN).**
10. **When the main menu reappears, choose option 4 to return to MS-DOS.**
11. **Type FORMAT C:/S (RETURN).**
12. **When formatting is complete, type CHKSK C: (RETURN) to check that the disk is correctly set up.**

To copy files to the hard disk, use the same techniques as for copying to floppy disks, but with the drive letter C:. It's likely that you will want to arrange files into subdirectories in the hard disk; this is dealt with in Part 4.

PART TWO

Simple DOS use

■ SECTION 24
Loading

The main and most important DOS action is the loading and running of programs, and this requires knowledge of the program filenames.

When you make use of GEM, the names are displayed on the 'icons', and you choose the program by using the mouse. GEM is not an operating system in its own right; it actually makes use of DOS Plus or MS-DOS in order to carry out the loading and running action. In a sense, then, GEM is an intermediate between you and the operating system.

When you make use of MS-DOS or DOS Plus direct, the program is loaded and run by typing its name and pressing the RETURN key – with the advantage that none of your memory has been taken up with GEM! The important point, however, is that you must know the names of your programs on the disk(s) that are present in the drive(s). This is done by using the DIR utility.

Whichever variety of DOS you use contains, along with its disk managing programs, several **utility** programs which will make life easier for the user. Some of these utilities are **internal,** meaning that they are read into the memory of the PC1512 when you switch on and insert the System disk. These utilities can be called into use at any time simply by typing the name of the utility that you want, with no need to have the System disk or any other disk in the drive. These internal commands are as follows:

BREAK – used to control action of Ctr-Break keys (see Appendix A).
CHDIR – used to switch directories (see Part 4).
CLS – clears the screen.
COPY – used to copy files.
CTTY – used when computers are connected together (see Part 7).
DATE – sets date.
DEL – used to delete a file or set of files.
DIR – used to display directory information.
ERASE – used to delete a file (like DEL).
MKDIR – used to make a subdirectory (see Part 4)
PATH – used to find files in a subdirectory (see Part 4).
PROMPT – used to make a new prompt in place of A> (see Appendix A).
RENAME – used to rename a file on a disk.
RMDIR – used to remove a subdirectory (see Part 4).
SET – set up a 'name' for later use (see Appendix A).
TIME – used to set clock.

TYPE – used to display file data on the screen.
VER – displays MS-DOS version number.
VERIFY – turns on/off checking of disks as they are used.
VOL – displays disk label name.
XCOPY – copies files between subdirectories (see Part 4).

One useful utility allows you to change the **current drive**, meaning the disk drive that will be used unless you choose otherwise. If you are using drive A and need to use drive B, then typing B: will carry out the change. The prompt sign will now be B> rather than A>. To change back, type A:. This drive change also applies to other drive letters, so that you can use C: to select the C-drive (hard disk or RAM according to the model of PC). This type of selection is permanent until you switch off, or change it deliberately.

Some other utilities are **external.** Any one of these will be fetched from the System disk when you type its name, and are erased from the memory immediately you have finished with them. We'll look at more of these utilities in more detail in Part 3.

SECTION 25
Using DIR

One particularly useful resident utility is **DIR,** which will present you with a listing on the screen of the disk directory. At this point, we'll assume the use of a floppy disk; as has been hinted the directory of a hard disk is likely to be considerably more complicated. To use DIR, type DIR (RETURN).

From the directory listing, you'll see the programs that are available to you. The type that you can load and run are the programs with the extension letters of EXE or COM . When you type the name of such a program so as to load and run it, it is not necessary to type the extension COM or EXE, though it will do no harm if you do so.

```
A>

 Volume in drive A is MSDOS
 Directory of  A:\

ATTRIB   EXE     CHKDSK   EXE     COMP     EXE     DEBUG    EXE     DISKCOMP EXE
DISKCOPY EXE     EDLIN    EXE     EXE2BIN  EXE     FDISK    EXE     FIND     EXE
FORMAT   EXE     GRAFTABL EXE     GRAPHICS EXE     JOIN     EXE     KEYBUK   EXE
LABEL    EXE     MODE     EXE     PRINT    EXE     RECOVER  EXE     REPLACE  EXE
SHARE    EXE     SORT     EXE     SUBST    EXE     TREE     EXE     XCOPY    EXE
       25 File(s)       2048 bytes free

A>dir

 Volume in drive A is MSDOS
 Directory of  A:\

COMMAND  COM     ANSI     COM     APPEND   COM     ASSIGN   COM     MORE     COM
MOUSE    COM     RTC      COM     SYS      COM
        8 File(s)       2048 bytes free
```

If, however, you are using the name of a file as part of a command to do something with that file (like copy it, rename it, delete it), then the *full* name, with any extension, must be typed. One of the advantages of using GEM is that you don't have to remember these points because they are all carried out automatically for you when you point with the mouse and press the mouse button.

Another point is that, if you are using more than one drive, you may want to specify a drive letter along with the name of the program that you want to load and run. As always, the drive will revert to the normal (usually A) when the program starts to run. Thus DIR B: will give the directory for the B drive, but the next command that you use will refer to the A drive unless you use B: in that command also.

(Further information: Amstrad Manual, page 250.)

SECTION 26
Wildcards

The term **wildcard** sounds very fanciful when applied to computing, but it's a useful feature of most disk systems. The wildcard is a character, usually *, that can be used as a substitute for a character or group of characters.

When you use the asterisk wildcard you are saying, in effect, that you don't care what that part of a name is. For example, if you type DIR *.COM and press RETURN, you will get a listing of any files that have the COM extension, whatever the main part of the filename happens to be.

If you type DIR MA*.TXT , then (if the disk contains them) you will get names such as MAINONE.TXT, MANALIVE.TXT, MARTIN.TXT, MALLARD.TXT, and so on. The use of wildcards can be very convenient, allowing a lot of actions to be carried out with just one command.

It can also be very inconvenient, causing you to delete a file accidentally just because you were deleting a group that happened to have similar names, for example. The asterisk is the wildcard that is most often used, because it's so convenient to be able to substitute for any number of characters.

There is also a single-character wildcard, the ? character which you can use if you want to allow uncertainty about only one character. For example, using DIR *.BA? would print out all the files that have the extension letters BAT and BAS , and any others that used BA as the first two letters of the extension.

Whatever you specify by using wildcards, the directory will always supply the date and time when each file was placed on the disk (see the next Section).

(Further Information: Amstrad Manual, page 94.)

■ SECTION 27
Date and time

The PC 1512 contains a digital clock circuit that operates from batteries, and so maintains the correct time and date whether the computer is switched on or not. The batteries are in the hatch on top of the disk-drive box and, if the computer is switched on, removing the batteries will not stop the clock.

The use of this system avoids the need to enter a date and time into the machine each time it is switched on, as was necessary with older types of PC machine, but you may want to change the date, and you will certainly need to change the time at least twice a year to allow for Summer time start and end. These actions can be controlled by the programs **DATE.COM** and **TIME.COM** which are internal utilities that are loaded into the memory each time the machine is switched on from cold. You do not therefore need to have the System disk in place when you use either of these utilities.

Either utility can be used simply to read the date/time or to alter it, and the .COM extension does not need to be typed.

(Further Information: Amstrad Manual, pages 292 and 480.)

■ SECTION 28
Date

To read or alter the date display:

1 **Type DATE and press the RETURN key. The screen will show the date, giving the day as a three-letter name (Mon, Tue, etc.),and the date in day-month-year order. Note that most machines in this class show dates in the US form, as month-day-year.**

```
A>
Current date is Thu 18-12-1986
Enter new date (dd-mm-yy): 21-1-1987
```

2 **If you simply press the RETURN key at this point, then the date will be unchanged.**

3 **To change the date, then enter the date that you want, using either the hyphen (-) or the slash (/) signs to separate the numbers. For example, you can enter 7 June 1987 as 7-6-87 or as 7/6/87.**

4 **When you have entered this date, which appears on the screen next to the words 'Enter new date', then pressing the RETURN key will make this the established date and the correct date will be maintained for as long as the clock runs.**

(Further Information: Amstrad Manual, page 346.)

■ SECTION 29
Time

To read or alter the time display:

1 **Type TIME, and you will see the time appear in hours, minutes, seconds and hundredths of a second.**

```
A>time
Current time is 13:17:51.30
Enter new time: 12.42.50
```

2 **As before, you can press RETURN to leave this as the correct time setting, or type a new time.**

3 **For most purposes, you will need only to type the hour number (0 to 23), a colon, then the minute number (0 to 59), and press RETURN at the time when the seconds will be zero, as indicated by a quartz clock or other source (like the TIME display on a Teletext TV receiver). This is preferable to trying to enter a number of seconds with any accuracy.**

(Further Information: Amstrad Manual, page 354.)

SECTION 30
DIR modifiers

The straightforward use of DIR produces a listing of all the files of data that are on a disk. You can specify which drive to use by adding the drive letters, such as A:DIR, or B:DIR and so on. For the use of C:DIR, see Part 4.

You can also add letters following the DIR command to specify how you want your directory displayed. These letters are typed following a slash symbol (/ on the ? key). The letters that you can use in this way are **P** and **W**.

Typing DIR/P will give the directory of the currently used drive, in 'pages'. The screen will fill with the names and details of the files, and then halt so that you can read the whole screen. The listing is continued when you press any key.

Typing DIR/W will show the names of the files only. The name includes the extension, but no details of file sizes or the date and time of creation will be shown. This is useful if you just want to check quickly that various files exist on the disk.

```
 Volume in drive A is MSDOS
 Directory of  A:\

IMAGES          COMMAND  COM     ANSI     SYS     CONFIG   SYS     DRIVER   SYS
RAMDRIVE SYS    AUTOEXEC BAT     GEM      BAT     GEM3     BAT     ANSI     COM
APPEND   COM    ASSIGN   COM     MORE     COM     MOUSE    COM     RTC      COM
SYS      COM    ATTRIB   EXE     CHKDSK   EXE     COMP     EXE     DEBUG    EXE
DISKCOMP EXE    DISKCOPY EXE     EDLIN    EXE     EXE2BIN  EXE     FDISK    EXE
FIND     EXE    FORMAT   EXE     GRAFTABL EXE     GRAPHICS EXE     JOIN     EXE
KEYBUK   EXE    LABEL    EXE     MODE     EXE     PRINT    EXE     RECOVER  EXE
REPLACE  EXE    SHARE    EXE     SORT     EXE     SUBST    EXE     TREE     EXE
XCOPY    EXE    TESTIT           PRINBOLD
       43 File(s)      2048 bytes free
```

You can get any of these directory displays listed on paper if you have a printer attached. After typing the command (DIR or DIR/W, for example), press the CTRL and P keys together. This combination is written as CTRL-P. When you press RETURN, the directory will be printed. Do *not* use CTRL-P unless you have a printer connected and ready.

To disengage the printer, press CTRL-P again. This is *important*, because if you do not, then every subsequent command and its effect will be printed. There is nothing to indicate in which state this command is at any time, so that if you have pressed CTRL-P, then it's up to you to remember to press it again.

■ SECTION 30
DIR modifiers

You can use the DIR command to get information on one file only, by adding the name of the file to the DIR command, such as DIR DATE.COM. One reason for doing this would be to find out how much memory space a program needs or how large a file of data is.

A much more common requirement is to report on a set of files with a common extension, so that, for example, DIR *.COM/P would list for you all the files with the extension COM, and page the results, since there will be many files of this type. Similarly, you could request the longer files that have the EXE extension by using DIR *.EXE/P – again, you would want to page this because a floppy disk would contain several such files. You might also have a very large number of files of this type on the main (root) directory of a floppy disk (see also Part 4).

(Further Information: Amstrad Manual, page 250.)

SECTION 31
Hard disks: introduction

The basic model of Amstrad PC1512 comes with a single disk drive, but for business purposes at least two floppy disk drives are needed, and most business users specify the use of a floppy drive and one hard disk.

The hard disk models come in two sizes, the 10 megabyte and 20 megabyte, referring to the storage capacity of the hard disk. One megabyte corresponds to 1,048,576 bytes of storage. As supplied by Amstrad, the hard disk replaces one of the floppy disk drives.

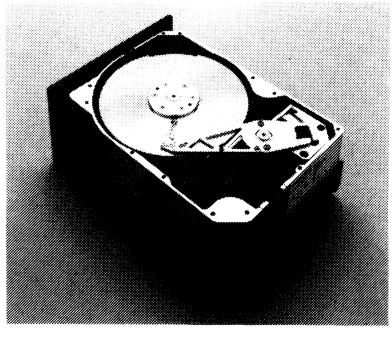
Some owners will prefer to buy the twin floppy drive machine and upgrade to hard disk by means of an add-on card. In this way, the benefits of both systems can be fully realised.

The hard disk, sometimes called a Winchester disk, is a way of obtaining a much larger amount of information packed into the normal size of a disk. The older name of Winchester was used because this was the IBM name for a project that called for the development of a hard disk in a sealed unit in the early 1970s.

Like other types of hard disk, Winchesters keep the disk itself and the magnetic read/write heads sealed inside a container. This ensures a dust-free environment for the disk. Unlike some other types of hard disk unit, however, the Winchester is *permanently* sealed – the disk is not removable, and the sealing will be disturbed only if repairs are needed – and that will require an air-conditioned workshop.

This sealing into a clean dust-free space allows the gap between the head and the disk to be made much smaller than could be contemplated for a floppy disk, smaller than a grain of dust or a particle of smoke. Technical leaflets refer to the head as 'flying' and this is not advertiser's talk – the head actually floats on a thin film of air between the head and the disk.

This very small gap between the head and the disk allows a much higher packing of information on to the disk so that the most obvious effect of using a hard disk is the much greater number of bytes that can be stored.



Hard disk unit

■ SECTION 32
Hard disk advantages

Unlike the comparatively slow-spinning floppy disk, which cannot be rotated fast because the disk rubs against its envelope, the hard disk spins at a very high speed, around 3000 revolutions per minute. This means that the rate at which data can be written to the disk or read from it is much greater, some six times as fast as a floppy disk.

This speed of recording or recovery of data has been helped by the progress in reducing the size of these hard disks. The first units used 14-inch disks and, by the later 1970s, 8-inch units were in full production. The modern 5¼-inch hard disk started to appear by the early 1980s, but it is only recently that prices have dropped to a level that allows really widespread use of hard disks on small computers.

The advantages of the smaller size have been lower costs, and faster access to data, because the head needs to move over a much shorter path on a 5¼-inch disk than on the 8-inch disk.

■ SECTION 33
Why use the hard disk?

Unless you have had previous experience of using a hard disk, it isn't exactly obvious what advantage other than sheer size of storage space they have. Since the Amstrad PC1512 manuals for the floppy-disk machines are not very informative about the use of the hard disks, a short tutorial seems appropriate here.

To start with, hard disks are a good solution to the frequently used program that takes up two or more floppy disks. For example, WordStar 1512, the word-processor program that is strongly recommended for use with the PC1512, comes on six floppies. Though in everyday use of the program you will not necessarily be inserting and replacing all six of these, it's nevertheless a considerable drag to have a program split up in this way.

By transferring all of the files on to one hard disk, the entire set of files becomes available to you as soon as you have switched your computer on. Even if we assume that each of the six disks is completely filled, this represents only about 2.1 megabytes, and the smaller hard disk model of the PC1512 uses a 10-megabyte disk!

From the time that the files are transferred to the hard disk, the files become accessible from the instant the machine is switched on, because the hard disk is built in as part of the machine. The files can also be accessed much more quickly, because there is no need to shuffle disks about, and the access time for the hard disk is so much faster.

You could therefore keep all of your business programs, word processor, spreadsheet, database and ideas processor, on one hard disk. Even more useful, you can keep all of your MS-DOS Master disk contents, and arrange for this to be loaded at the time of switch-on, and still leave room for data if you wanted to do so. For a lot of purposes, though, its better to use the hard disk for programs, and the floppy disks for data apart from data that is unchanging and which needs to be used frequently.

With the larger hard disk, of course, there is ample space for data as well as for program files, but you will still probably prefer to keep data on floppies, if only from the point of view of having several backup copies.

■ SECTION 34
Hard disks in service

Making regular use of a hard disk is quite different from using floppies. For one thing, the hard disk spins continually as long as the machine is in use, so as to cut down access time.

Much of the time that you spend waiting for data to be read from a floppy disk is spent in waiting for the drive motor to spin the disk up to its correct speed. The important point to remember is that you can't easily replace the hard disk, so that if you ever fill it, you will have to delete a file in order to record another file.

This is not exactly likely to happen within the first few weeks of putting a machine into service, but you need to plan your use of a hard disk rather more carefully than you do with floppies.

You need to consider, first of all, what programs you want to put onto the hard disk and how you are going to go about this. We shall look at file copying commands in Part 3, and consider in Part 4 how the PC's provision for dividing up directories can be applied to hard disks.

■ SECTION 35
Hard disk backup

The backup of a hard disk is of very considerable importance, because though the destruction of a floppy disk can cause you to lose up to 360K of data, the destruction of data on a hard disk can result in the loss of 10M or 20M according to which model you are using. This amount of data loss can be completely crippling, and if not backed up would spell the end of most businesses.

Backup is no problem if programs have been bought on floppy disk. Unless you have been unwise enough to buy programs that are copy-protected, you will have transferred these programs to the hard disk and retained the original floppies as backup, perhaps making an additional set of backup floppies.

The loss of a hard disk containing only such programs is therefore not quite so serious as the loss of a disk that contains data. If you keep, for example, all of your day-to-day accounts data on the hard disk, with no backup, then the loss of the disk will spell ruin. All data files on the hard disk should be backed up onto floppy disks, even if this means keeping a stock of several hundred floppies (you need about 29 floppies to back up the data on a 10M hard disk).

The price of floppy disks in packs of 100 is now so low that the price of the disks is negligible in comparison to the value of the data. The more important factor is the time needed, and it helps if programs that you use will make one copy of data on to the hard disk and another on to a floppy disk at the time when the data is created or altered. In this way, the backup is made in a more-or-less painless way.

SECTION 36
Complete backup

If regular backing-up has not been practiced, then the alternative is the backing up of a complete hard disk. On the Amstrad PC, you would normally use floppies to do this, because there is always a floppy drive available.

This is not the only method, and machines exist that can be connected to the PC and which will back up a hard disk onto tape cartridges or onto videotape cassettes. These have the advantage of speed, because backing up the contents of a 10M hard disk on 29 floppies is not by any stretch of the imagination a fast process.

The problem at the time of writing is the high cost of these 'tape streamers' and video cassette machines as compared to the PC1512 itself. It's possible, however, that cheaper methods may be on the way, particularly when the new 8-mm video tape standards start to be used in the new generation of video and audio recorders.

■ SECTION 37
Damage to hard disks

The most common cause of damage to a hard disk is physical knocking that causes the drive head to come into violent contact with the disk. The early types of drive left the head placed over the disk when the machine was switched off, and this meant that picking up the machine and putting it down again could jolt the head against the disk, denting the disk.

The risk is much greater if the machine is jolted while the disk is being used, because in normal action, the head 'flies' over the disk with a separation of only a few millionths of an inch. A jolt while the machine is operating can therefore cause a track to be cut in the disk, causing irreparable damage. Modern hard disk units fold the head out of the way when the machine is switched off, so that the machine can be transported safely, and the PC1512 is very rugged in this respect.

Any hard disk machine, however, is very susceptible to being jolted when the hard disk drive is in use, and some care should be taken over this. If a disk is damaged in this way, the whole hard disk unit will have to be returned for repair (and insertion of a new disk), and unless you can disconnect the unit yourself this can mean losing the use of the whole computer – not a pleasant prospect if you rely on it for your business.

If you keep a complete set of backups on floppy disks, then you can get the hard-disk unit removed and sent away, leaving you a computer with a floppy disk drive that you can still use, albeit at a slower rate. An alternative is to keep a spare hard disk unit in the form of a plug-in 'card'.

Damage like this is the main cause of premature failure of a hard disk unit. When smaller hard disks were first introduced, the reliability was being quoted as about 10,000 hours mean time between failures. This corresponds to about 3.5 years of very hard use, and unless you are remarkably unlucky, this means that the life of the disk will be as long as you need, since it is normal to write off the computer in this time.

In fact, the lifetime should be considerably better than this type of estimate comes up with, because the mean time between failures is worked out on the basis of much more intensive use than you are likely to give the hard disk, even in office use.

■ SECTION 38
Backing up commands

MS-DOS has two commands that apply only to machines that are fitted with a hard disk. These commands are not present on the MS-DOS distribution disk for floppy-only machines. The first of these is **BACKUP** which is used to transfer all files on a hard disk to a set of floppy disks. To use BACKUP for a single file:

1. **Insert a fresh, formatted floppy disk in the floppy drive that is currently being used.**
2. **Type BACKUP C:hardfile A:, where hardfile represents the name of a data file. Press RETURN.**
3. **Obey any messages that appear on the screen about changing floppy disks.**
4. **Remove each floppy disk in turn, label it, and put on a write-protect tab. It is important to write a series number (1, 2, 3,...) on each label in turn if more than one disk is needed. This is because the disks must be used in the same order when replacing data on to the hard disk.**
5. **If you need to back up the whole hard disk, then use C:*.* as the filename, and have sufficient floppy disks ready, formatted, and numbered in sequence. Remember that you need 29 floppies to back up a 10M hard disk completely.**
6. **If you need only to back up data, then use a suitable wildcard in the filename, or copy files on an individual basis as above.**

SECTION 39
Returning files

Files backed up from a hard disk on to floppy disks can be restored to a hard disk by using the **RESTORE** command. Note that this is *not* used for copying files to hard disk, only for restoring the state of a hard disk that has lost data. In other words, if the floppy disks that you use in RESTORE were not created by the BACKUP command, then they will be rejected.

To use RESTORE:

1. **Place the first of the backup disks in the drive A.**
2. **Type RESTORE A: C:hardfile where hardfile represents the name of a hard disk file to be restored. Press RETURN.**
3. **You will be prompted to replace floppy disks as needed. The disks must be in the correct sequence, as noted under BACKUP.**
4. **To restore all the contents of the hard disk, use the RESTORE command with a wildcard filename *.***

PART THREE

Internal commands

SECTION 40
Renaming a file

The need to rename a file arises more often than you might expect, and there are several reasons why this action should be needed.

One common reason is running any program that makes automatic use of the BAK extension when a file on a disk is replaced by another of the same main name. MS-DOS does *not* do this, but many word processors do. Suppose, for example, that you have just made a lot of changes to a file called MYTEXT.TXT, and then saved the file back to the disk.

This will result in the older copy, the one that has been changed, becoming automatically renamed to MYTEXT.BAK, and the previous version, that had been called MYTEXT.BAK up to that moment, will be wiped off the disk completely.

Suppose now that you want to refer to what was in the text that you have just altered? A lot depends on whether the program that edited the text will allow you to read a .BAK file. If it does, there is no problem, you read in the MYTEXT.BAK file, do whatever you want to do with it, and then save it under another name.

This *can* be time-consuming, however, and not all programs will allow .BAK files to be read. The simple alternative is to rename this file to a name that *can* be read, like OLDTEXT.TXT. To do this requires the use of the **RENAME** command, which can be shortened to **REN**. REN is an internal command, so that you do not need to have the MS-DOS disk in the drive at the time when you press RETURN.

The form of the command is

REN oldname newname

with a space between the old name and the new name. The filenames can be complete, such as A:MYTEXT.BAK , or the drive letter can be omitted, in which case drive A will be used, or whatever is the drive in use.

SECTION 40
Renaming a file

```
WS1dir

 Volume in drive A is 1512 SYSTEM
 Directory of  A:\

COMMAND  COM     23612  14-07-86  12:13
TEST     DOC        34  18-12-86  13:54
        2 File(s)     290816 bytes free

A>
A>ren test.doc whatsup.doc

A>
A>dir

 Volume in drive A is 1512 SYSTEM
 Directory of  A:\

COMMAND  COM     23612  14-07-86  12:13
WHATSUP  DOC        34  18-12-86  13:54
        2 File(s)     290816 bytes free
```

If you are using a hard disk, you will probably have to specify more information in the filename to show in which directory the file is located. This problem is dealt with in Part 4.

In the example of a floppy disk file, the renaming would be carried out by using REN MYTEXT.BAK OLDTEXT.TXT, and pressing the RETURN key will carry out the change of filename.

(Further Information: Amstrad Manual, page 315.)

■ SECTION 41
Wildcards

You may also have to rename a file that is not a .BAK file, but a current one used by a program. Many programs will create files and automatically assign a filename to data so that the data can be obtained again when the program is next used, without the need for the user to specify a filename.

Another use for RENAME is that the RPED text editor, included on your Disk 3, will not accept a name for a new file that includes an extension. You often need to rename such files after having created them.

If for any reason you want to *prevent* such a file being read and used, you will have to rename it. It's more likely that you will want to make a copy under another name, however, and that's a topic that we'll come to later in this Part.

If you have a large number of files to rename, then a wildcard can be used. For example, you could force all .TXT files to be renamed as .WP files by the command:

```
REN *.TXT *.WP
```

and then rename all .BAK files into .TXT files by using:

```
REN *.BAK *.TXT
```

This type of command has to be used with some care, however, because it can have rather drastic effects on your files. If, for example, you have a number of files with the extension TXT and you command the renaming of all .BAK files into .TXT files, what happens to any .TXT files that happen to have the same main name as the .BAK versions?

As it happens, MS-DOS provides for this, and the renaming is *not* carried out. Instead, you get a message to the effect that a duplicate file name has been specified, or the file has not been found.

Before you make use of any wildcard in a command like REN, you should study a *printout* of your directory to make sure that nothing unexpected will happen.

(Further Information: Amstrad Manual, page 315.)

■ SECTION 42
The TYPE utility

TYPE is another internal command, which is available without the need for a System disk to be in drive A. Its use is mainly on files that consist of the data output from word processors, spreadsheets and databases, in other words, the standard office programs.

TYPE should *not* be used on a program – any file with a COM or EXE extension, for example. The reason is that TYPE places on the screen the shape corresponding to each number-code in a file. Now if the file is a file of text letters, the numbers in the file are numbers of the standard **ASCII** code. The letters mean American Standard Code for Information Interchange, and this is what ASCII code is – a number code for each of the letters of the alphabet, digits from 0 to 9, and punctuation marks, plus a few actions like TAB, backspace and beep.

If a file on your disk consists only of ASCII codes in the range 32 to 127, with perhaps a few in the range 0 to 31, then each number code in the file will correspond to something that can be displayed on the screen.

A file like this is called an ASCII file, and practically all programs that generate data will have some provision for saving data in this handy and universal form. Any data that has been saved in this way can be displayed on the screen by using TYPE.

```
A>type whatsup.doc
This is a file, would you believe.
A>
```

For example,

TYPE B:NEWFILE.TXT

will result in the text file of this name being read from the B drive and displayed on the screen. Since TYPE is an internal utility, this saves the time that you would otherwise need to find the disk that created the file and loading in a program to read the file again.

(Further Information: Amstrad Manual, page 322.)

■ SECTION 43
TYPE in pages

One common problem in using TYPE is that the text scrolls continuously on the screen. One way of controlling this is to use the CTRL-S key to stop the scrolling, and the CTRL-Q keys (or *any* single key) to restart. This is useful, but clumsy if you want to read a long file.

Another solution is to print the file, pressing CTRL-P before pressing RETURN on the TYPE command, and again after all the file has been printed. Use this option *only* if you have a printer connected and switched on.

There is a method, poorly documented in the manual, for obtaining output from TYPE, one screen at a time. This uses some commands that we have not yet looked at, but these will be explained later. The procedure is:

1 **Place the MS-DOS System disk in drive B, and the disk containing the text in drive A. If you are using a single drive, you start with the disk whose text you want to read in drive A, and swap disks as requested.**

2 **Type the command line:**

TYPE A:TEXTFILE.TXT | B:MORE > CON

Make certain that this has been correctly typed. The | symbol is on the key to the left of the letter Z. Be careful about spaces.

3 **When you press RETURN, you will see the file appear one screenful at a time. Press any key to obtain the next full screen.**

4 **Press CTRL-Break if you want to stop viewing a file before the end.**

SECTION 44
TYPE limitations

TYPE cannot be used with any success on a file that contains codes other than the standard ASCII range. Programs, other than those written in BASIC 2, are all of this type, and many programs will create files that are of this type also.

For example, when WordStar saves a file of text, it saves the text in a specially coded form, with the code for the last letter in each word specially altered, and the text also includes many other special codes. Any attempt to read text that has been saved by WordStar *in the ordinary way*, using TYPE, will cause some very odd effects. Most word processors behave in a similar way, because a lot of disk space is saved when text is recorded in this coded form.

```
A>

 Volume in drive A is p`
 Directory of  A:\

}o 1611989312  17-00-96   0:08
 @721592361   2-08-99  12:01
WY [ @J4278624361    6-08-31  12:03
                  @      2869600425  10-08-63  12:05
■3043604457  14-08-95  12:07
!Aa  <DIR>      17-08-96   2:08
/A722641193  18-08-99  12:09
        7 File(s)       3072 bytes free
```

All word processors, however, make provision for text to be saved as an ASCII file, one that TYPE *can* read, and also to read such text. A number of word processors do save their text as ASCII files, and this also applies to spreadsheets and databases and other business programs that generate text or similar displays.

If you want to make effective use of TYPE, then you have to make sure the programs that generate data are saving a copy in ASCII code form as well as in their own codes. This may mean making two copies of everything, and in such cases you might feel that it wasn't worthwhile just so as to be able to use TYPE.

■ SECTION 45
Documentation

One other use for TYPE is to read text that is supplied as documentation. Sometimes a program consists of a number of files, one of which is a set of last-minute instructions, bearing a filename such as READ.ME. If you find such a file on a program disk, you should print out this file right away, because the information in it is likely to be an upgrade or correction of the manual, and reading this file can save a lot of frustration in trying to use a command that doesn't appear to work the way that the manual claims it should.

Some programs, in fact, contain no documentation other than the READ.ME or .DOC type of file. This applies particularly to public-domain and user-supported software (*Free* software) which dispenses with paper documentation because of the cost of distribution.

```
NY

 Volume in drive A is #529 V1
 Directory of  A:\

HISTORY  DOC      3382  14-12-85   16:27
INSTALL  DOC      3746  29-05-86   13:56
INTRO    DOC     18085   2-06-86   23:57
REGISTER DOC       510  14-12-85   16:26
        4 File(s)     117760 bytes free

A>

 Volume in drive A is #529 V1
 Directory of  A:\

GO       BAT       793  11-06-86    9:54
PRINT    BAT      1563  23-05-86    8:51
        2 File(s)     117760 bytes free
```

To print out the READ.ME file, make sure the printer is connected, loaded with paper, and switched on. After typing the TYPE command, press the CTRL and P keys of the PC1512 as noted above, then press RETURN to request a print of everything that would be shown on the screen.

The document will be displayed on the screen *and* printed on paper. At the end of the printout, remember to press the CTRL and P keys again to turn off the printer action, otherwise each command that you type will cause more material to be printed. If the document is too lengthy, you can stop the printing action at any time by pressing CTRL-P again. The listing will continue on the screen, but will no longer be printed. To stop the action *completely*, use CTRL-Break.

■ SECTION 45
Documentation

Note that the TYPE command does not permit the use of any wildcard characters, so that the filename must be completely specified. This is particularly important if you want to TYPE a file from a hard disk, because you have to specify the directory path (see following Part) for the file as well as the main filename.

In a very few cases you can glean something useful from a program file of the .COM or .EXE type. Most of such a file will cause the screen to display rubbish, to clear, flash, change colour or produce other unwanted effects.

Many programs, however, contain copyright notices and other *hidden* messages in ASCII code which do not necessarily appear when the program runs normally. It can often be useful to use TYPE on the first part of a program file just to see if any such messages are held there.

You can sometimes be rather surprised to find just how old a program can be, or to find who the original programmer was, or for what machine it was originally written. Other than this, however, TYPE has no real application to program files.

SECTION 46
The COPY command

The **COPY** utility is yet another internal program whose use does not require the System disk to be present in drive A. As the name suggests, COPY can be used to make a copy of a file on the same disk, using another filename, or on another disk.

The use of COPY is considerably more extensive than this, however, and it has a lot of applications in batch files, as we'll see in the following Part. For the moment, though, we'll confine our attention to the simpler uses of COPY. Like the DIR command, COPY permits modifications to be made to the action of the command by adding letters separated by a slash sign, and it's the use of these extensions to the command that take some time to become familiar with.

The simplest use of COPY is in the form

COPY source destination

with the filename of the file you want to copy following the command word, and the filename for the copied file following, separated by a space. As you might expect, you can use a full filename (with drive letter, main name and extension) for both source and destination (and hard disk users will also have to provide directory paths).

For example, using:

COPY A:MYFILE.TXT B:NEWFILE.DOC

will make a copy of a file called MYFILE.TXT which is on the disk in drive A, and create the copy under the filename of NEWFILE.DOC on the disk in drive B. If you typed simply:

COPY A:MYFILE.TXT B;

then the copy would use the same filename.

You are allowed to use a wildcard symbol in either the main name or the extension part of either filename, but if you are making another copy of a file on to the same disk, you are *not* allowed to use the same filename for obvious reasons.

You can also use the wildcard name *.* in the source file, so as to copy all files to another disk. This command must be used in the form:

COPY A:*.* B:

because you can't copy a set of files to the same disk with the same names. If you try to do this, you will get the error message 'File cannot be copied onto itself'.

SECTION 46
The COPY command

The hard disk user, however, or the user of a floppy in which a tree-directory is applied, can transfer a file onto another directory using the same name.

(Further Information: Amstrad Manual, page 299.)

SECTION 47
Joining files

One extension to the simple use of COPY is of particular interest to anyone who uses a word-processor program. It's normal to write and save text in small chunks, perhaps a thousand words in each chunk, so as to ensure that there is no chance of losing too much text because of a momentary interruption in the electricity supply. Some word processors even save text automatically after each 2000 characters have been typed, corresponding to about 330 words.

These small chunks can be combined in a COPY command, simply by typing the filenames in the source part of the command, separated by + signs. For example, if a chapter of a book consists of files CH4-1, CH4-2, CH4-3 and CH4-4 , then the command:

COPY CH4-1 + CH4-2 + CH4-3 + CH4-4 NEWCH4

will combine all four sections into the file NEWCH4, a much longer section of text. If the filenames have extensions, you *must* include these, or use .* as a wildcard extension. You cannot use a wildcard as the main name.

Before you go overboard about this, however, check that your word processor program can cope with longer files. It's most unlikely that any self-respecting word processor could not cope with a file of four thousand words or so, but I can't guarantee that you are using a self-respecting word processor!

Programs like WordStar use the disk as **virtual memory**, which means that the text is held on the disk except for the part that is being worked with, so that the length of the text file on the disk is not a problem.

A few word-processing programs, however, rely on reading a complete file into memory and working on the file in memory. For such a program, you would have to be certain that joining up your files in this way did not make any file too large to work with, because it's a lot easier to join up files than to separate them again.

Fortunately, this type of problem is rather rare now, and the memory of the PC is more than adequate in any case. Normally you would join up files only for the purposes of printing them with consecutive page numbers, and you would have kept the sections separately. Even word processors that keep text in memory for editing purposes will normally read the text from disk when it comes to printing.

(Further Information: Amstrad Manual, page 300.)

SECTION 48
Other extensions

The other extensions to the COPY command are more specialised, and they assume that you know something about the type of file that you are copying. The point is that there are two types of file that you are likely to have on a disk.

One type is files of text, including ASCII files. These consist of codes that are in the ASCII range, or an extension of the range, and the feature that they have in common is an end-of-file marker. This is the code number 26 which is placed on the disk *automatically* by the program that has created a text file, as an indication that this is the end of the file.

When a file of text is read, for example by the TYPE command, the reading continues until this marker is found. If a COPY command uses the extension /**A** following the name of the source file, this indicates that the file will be read up to the marker, but not beyond it *even if the file is longer*. The point here is that some programs record text with an end-of-file marker, but then place other codes following the marker.

The normal file-reading actions of the computer will read all the codes in a file, irrespective of any end marker. By specifying the /A extension, you copy only the ASCII part of a file that might contain other codes following the end-of-file marker.

If you add the /A extension to the destination name, then the end-of-file marker will be placed at the end of the copied file, ensuring that the end of this file will be correctly recognised by word processors, TYPE and other utilities. For example,

```
COPY A:OLDTXT.TXT/A B:NEWTXT.TXT/A
```

would make a copy of the file OLDTXT from drive A, reading the file as far as the end-of-file marker, and copy this to drive B under the name NEWTXT, adding an end-of-file marker to it. The use of /A on a text file should not really be necessary, but is a useful precaution.

(Further Information: Amstrad Manual, page 300.)

■ SECTION 49
Other files

Not all files make use of an end-of-file marker, and in some cases it would be a disaster to stop reading at the marker or to insert one.

The reason is that some files, particularly program files, are quite definitely *not* ASCII files, and they could have the code number 26 as part of their codes quite by chance. For such a file, you need to read right up to the last number in the file as determined by the file-length number that is stored as part of the directory for the disk.

You also need to be sure that no end-of-file marker, code 26, is added to the file, because this could cause the program instructions to be altered in such a way that the program would no longer operate. Files of this type are called **binary files**, and reasonably enough, the extension letter that is used is /**B** . By specifying, for example:

```
COPY BINFILE.BIN/B PROG2.BIN/B
```

you read the binary file BINFILE.BIN from the current drive (usually drive A) and make another copy on the same drive under the name of PROG2.BIN. Because of the use of the /B extension, the first file is read right to the last code in the file, and the copy is made with no end-of-file codes.

The COPY command can also be used in quite a different way, as a method of transferring files from one device to another, such as keyboard to disk, disk to screen, serial input to disk, keyboard to serial output and so on.

This is a rather different type of use of the COPY command, and it is covered in more detail in Part 7, along with the principles of pipes and filters – all principles that make MS-DOS particularly useful once you have had some experience in its use.

SECTION 50
Erasing a file

Sometimes a file has to be erased or deleted, making use of the **ERASE** or **DEL** commands of MS-DOS. The effect of erasing a file is to alter parts of the file directory, rather than to wipe out the actual information. This makes it possible to recover a deleted file, but not unless you have considerable experience. If you delete a file that you later realise you need, then recovery will be possible only if you have recorded nothing else on that disk since deleting the file.

You should make a backup copy of that entire disk, and keep the original until you or a skilled MS-DOS user can recover the file. If the file was lost because of any other problem, like switching off the machine before a file was correctly closed, it's possible that recovery could be achieved with one of the MS-DOS utilities **(RECOVER)**, but only if the file was a text file.

Recovery following the use of ERASE/DEL, or recovery of a program file after magnetic damage to the disk, is not nearly so easy, and requires the use of utility programs that are not provided on the System disk.

The form of an erase/delete command is

ERASE filename

or

DEL filename

providing the full filename in each case, though if you omit the drive letter, the current drive will be used by default. This type of use of ERASE or DEL is straightforward, and less likely to cause trouble than its use with a wildcard.

When a wildcard is used, it can be used either in the main filename or in the extension, or both. It's possible, therefore, to delete *all* the file contents from a disk by using DEL *.* , and though you may think you would never do such a thing accidentally, it is surprising just what is possible when you work at it. As a safeguard, you will be asked by a screen message if you *really* mean it when you use such a command.

A small trip of the little finger can wipe out a set of files with astonishing speed, so that you become very grateful that you kept a backup of each disk – you *did* keep a backup, didn't you?

■ SECTION 50
Erasing a file

A disk full of valuable files should be protected by placing a write-protect tab around the slot in the side of the disk envelope. If you have one or two particularly valuable files on a disk, you can protect them by using the utility **ATTRIB** as follows:

1 **Place the MS-DOS Master disk in drive A, and the disk that contains the file you want to protect in drive B, or at hand if you have only one drive.**

2 **Type the command:**

ATTRIB +R B:FILENAME.EXT

using the drive letter B and the full filename of the file you want to protect. Press RETURN.

3 **If you have one drive, you will be prompted to change disks.**

4 **The effect of this form of ATTRIB command is to protect the file so that it can only be read, not written to nor erased.**

5 **To check for such ATTRIB changes, type ATTRIB B:*.* (RETURN). This will give a list of whether files are protected (R) or not.**

The best advice, however, is *never* to use ERASE/DEL with a wildcard character unless you have a printout of the disk directory and can see exactly what the effect of the wildcard will be.

(Further Information: Amstrad Manual, pages 302, 294.)

PART FOUR

More advanced work

■ SECTION 51
Batch commands

The aim of achieving some mastery of MS-DOS is to allow you to exercise much more control over what your computer does. Up to now, you have used the computer DOS commands directly, typing a command and then pressing the RETURN key so that the command is carried out at once.

This type of use is often called interactive, and the alternative is called batch use, in which you issue a set of commands and then let the computer get on with the job of carrying out each one in sequence. Obviously, the commands have to be typed in rather a different way in order to achieve this, because what you are to do is write a miniature program of commands that are intended to be executed later and in a particular order.

This is done in MS-DOS, much as it is done in other operating systems, by writing a file that is recorded on to a disk. This file is called a **batch file**, and by typing the name of such a batch file, you can make the computer carry out the commands that are contained in the file. If you have previously used batch files with an older CP/M machine, you will find that the whole subject is a lot easier with MS-DOS.

Suppose, for example, that you had a batch file recorded on a disk under the filename BAKELIM.BAT. We can imagine that the purpose of this batch file is to delete certain files with a BAK extension, but only certain files. If we wanted to eliminate all files with a BAK extension, it would be much easier just to use DEL *.BAK , but anything that involves picking and choosing along with repetition is an ideal candidate for a batch file.

For the moment, we'll ignore what the content of this particular batch file might be, because the important point is that to carry out this elimination of the .BAK files, all you need to do is to place the disk in the drive and type BAKELIM, the name of the batch file. You might, of course, if you have twin drives, keep all of your batch files on one disk that you place in drive A and make the commands operate on files that are kept in whatever disk is placed in drive B.

(Further Information: Amstrad Manual, page 239.)

SECTION 52
Creating a batch file

A batch file is simply a set of commands, recorded in ASCII codes. You can create batch files with the **EDLIN** editing program that is on the System disk. This, however, is not an easy program to use, and the **RPED** program that is supplied on Master Disk 3 (Green disk) is very much easier to use.

Another method is to prepare the text of a batch file with a word processor like WordStar which allows you to create ASCII files (called non-document files in WordStar). The batch file consists of the usual commands of MS-DOS, plus a few very useful commands that are peculiar to batch files, and which make the batch file much more like a programming language in its own right.

Yet another method is to type

COPY CON filename

which will save everything you type to a file, using whatever you typed as the filename, when you type CTRL-Z and (RETURN) at the end of the file.

In this Part we'll concentrate on the use of RPED, because it is simpler to use than EDLIN and has no disadvantages as compared to EDLIN. Before we take examples of the creation of simple batch files, however, there are a couple of points that need to be noted.

One is that each batch file should have the extension letters **BAT**, since this identifies it as a batch file when you look at the disk directory. The fact that the file has this extension also allows the machine to locate the file when you type the main filename. If you do not use the extension BAT, then the file will simply be treated like any other file of text, not as a batch file.

The other point is that there is a very special reserved name for a batch file, **AUTOEXEC.BAT**. If you have a file of that name on any disk which contains the DOS tracks (a start-of-day disk or boot disk), then whatever is contained in this batch file will be carried out *automatically* after the machine has read the DOS and before it is ready for use by you.

This can save a considerable amount of time if you had previously started a computing session by issuing a set of commands for such purposes as, for example, preparing the printer for use, clearing a disk, copying, renaming or deleting files and so on. By putting all of these actions into an AUTOEXEC.BAT file, you are saved both the effort of ►

■ SECTION 52
Creating a batch file

carrying out the actions and the equal effort of remembering which actions are needed for which disk.

You might, for example, want all of your word-processing disks to carry one form of printer setup , and all your spreadsheet disks to carry another. It's likely, too, that any programs that you buy will have AUTOEXEC.BAT files, and you may want to look at these files (using TYPE) and possibly modify them to suit your own purposes. Part 7 contains examples of the use of AUTOEXEC.BAT.

(Further Information: Amstrad Manual, page 243.)

■ SECTION 53
Starting with RPED

To make use of RPED, you really need to have this program on the disk on which you want to create some batch files. For the sake of illustration, I copied the MS-DOS Master disk and deleted the following files:

DEBUG FDISK GRAPHICS MODE

all of which are .EXE files and not immediately needed. Deleting these files leaves plenty of room for RPED and any batch files we might want to experiment with. The following examples have all been carried out on this disk, referred to as the DEMO disk.

To place RPED on this DEMO disk:

1 **Type the command:**

COPY B:RPED A:

2 **Place the MS-DOS DEMO disk in drive A, and Master Disk 3 in drive B, if you have one. As usual, you will be prompted to change disks if you have one drive.**

3 **Press RETURN to carry out the copying. Check the directory of the DEMO disk to ensure that RPED is now present.**

■ SECTION 54
A simple batch file

In this and the following section we shall see how a simple batch file can be constructed and used. The batch file will delete some files from the DEMO disk, and then provide a label name for the disk. The label name allows you to identify the disk even if a sticky label has dropped off.

The files that will be deleted are EXE2BIN.EXE and GEM3.BAT, and the disk will be labelled with the name DEMO. Obviously, this can be done only once with each prepared disk, but the idea is to demonstrate, and a simple demonstration like this is easier to carry out than one which might involve a lot of preparations.

The commands that have to be used are DEL EXE2BIN.EXE, DEL GEM3.BAT and LABEL DEMO. Each of these is a conventional MS-DOS command, and we have used DEL previously, though **LABEL** is new. The form of LABEL is straightforward, and the only comment needed is that the name must not exceed 11 characters in length. We start, then by running RPED.

SECTION 55
Starting RPED

Like any other .EXE program, RPED is started by typing its name, RPED (RETURN), with the DEMO disk in the current drive (usually A). This brings up the RPED menu:

```
This screen editor is for small files (up to 750 lines) and uses
     normal Cursor, Page, Home, End, Insert and Delete keys.


                    f1 = Edit Existing File

                    f2 = Re-edit Previous File

                    f3 = Create New File

                    f4 = Quit
```

To create a new file, we need to use the choice selected by pressing the F3 key on the pad at the left-hand side of the keyboard.

You will then be asked to provide a name for the file. This is where new users can encounter difficulties. The name that you eventually use *must* have the extension BAT if it is to be used as a batch file. RPED, however, will reject any filename that includes an extension. This is so that a file cannot be used as a batch file until you are certain of it, and have renamed it. There is, however, no message to this effect.

For the sake of demonstration, use the name DEMO (you can use B:DEMO if you want the file on another disk in the B drive). When you press RETURN, you are ready to enter text. The position of each character that you enter is marked by a flashing cursor bar. The top line is a reminder of some editing keys:

```
This screen editor is for small files (up to 750 lines) and uses
     normal Cursor, Page, Home, End, Insert and Delete keys.
```

and some of the usual DEL keys on the keyboard are still active. DEL-RIGHT operates as normal, but DEL-LEFT simply backspaces without deleting. To backspace and delete, use the DEL key on the number keypad.

■ SECTION 55
Starting RPED

1. Type the command DEL EXE2BIN.EXE , and press RETURN. This enters the command in one line.
2. Similarly, enter the other two commands of the batch file, DEL GEM3.BAT and LABEL DEMO.
3. Check your file for accuracy. In particular make certain that commands and filenames are correctly spelled. You can move the cursor with the arrowed keys on the number keypad at the right-hand side of the keyboard.
4. When you are satisfied with the text, press the ESC key to record the file.
5. Press F4 to leave RPED.

■ SECTION 56
Using the file

You should now have a file on the disk, with the filename of DEMO. If a previous file had the same name, then the older file (if created by RPED) will have been renamed DEMO.BAK. Since RPED will not create new files with an extension, we now have to rename the file.

Type REN DEMO DEMO.BAT, and press RETURN. This will rename the file, and you can check that this has been done by using DIR. Check also that you have the files EXE2BIN.EXE and GEM3.BAT that will be deleted. You can also check for a label by typing VOL (RETURN).

Now type DEMO (RETURN). You will see the commands appear on the screen just as if you had typed them at that moment. You can then type DIR (RETURN) to check the directory. This should show the label name at the top (use DIR/P to stop this from scrolling out of view), and you will find that the two named files have been deleted.

If the file does not operate correctly, check its content by using TYPE DEMO.BAT. If you need to alter the file, load RPED again, and select the F1 option to edit an existing file. You need to specify the *full* name of DEMO.BAT here. You can then use the editing facilities of RPED. Remember that you press ESC to record the edited version and CTRL-Break (not just Break) to leave without recording the new version.

SECTION 57
Multiple .BAT files

There are no restrictions on the number of .BAT files that you may have on your disk (subject to available space) or how you make use of them provided that the commands are valid. If some commands are not valid, then you will get an error message, but this does not necessarily stop the batch file.

For example, if you repeatedly use batch file DEMO on the same disk as you tested it on, then you will get 'File not found' for each of the DEL commands, but the LABEL action will be repeated, and the whole batch file is carried out.

It can be very useful to keep several batch files on a disk, so that you have choices of actions. Since the filenames can refer to other drive(s), the batch file disk need not necessarily be the disk that is used to hold the files that are operated on.

If you use several batch files on one disk, you should try to make the filenames remind you of the action. For example, names like SETPRN, SCRNCOL or CHARLIN can remind you of batch files that set the printer (margins, bold face, 40 characters per line for book or magazine listings, for example), the screen colours for foreground and background, and the number of characters per line on the screen.

Though you can have as many .BAT files as you please, subject to overall memory, you can have only one AUTOEXEC.BAT file. This, and other more advanced batch file topics, are dealt with in Part 7.

SECTION 58
Directory trees

Directory trees are the MS-DOS method of making life tolerable for the user of the hard disk. The principle is to subdivide the directory system so that using DIR does not result in page after page of file listings. By subdividing the directory, files can be kept in groups and you can call for a directory of one group only, greatly reducing the effort that you need to spend on finding anything useful.

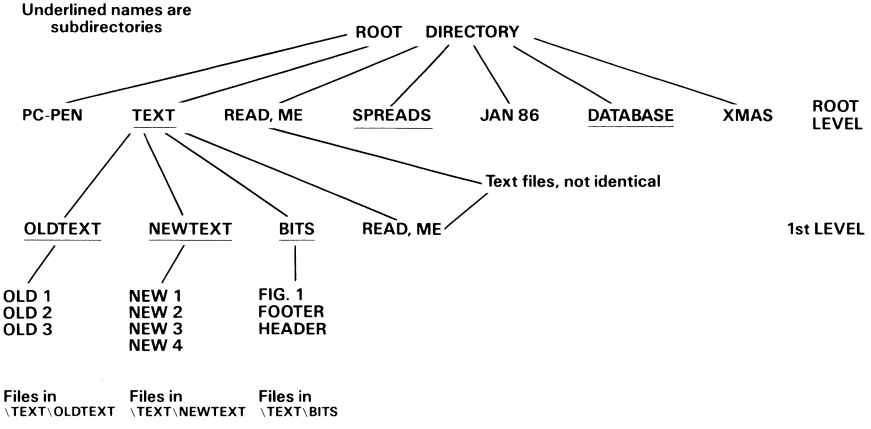
The scheme can be operated just as easily with floppy disks, but it is less necessary, because the storage capacity of a floppy disk is so much less than that of a hard disk. It's easier to use a separate floppy disk or group of floppy disks for a set of related files than to work with directory trees, though you will find directory trees on your Master disk set, particularly on the GEM disks 2 and 3.

The meat of this and the following sections, then, is mainly of interest to the user of a hard disk, and to the user of floppy disks who is likely to add a hard disk card to the machine in the future. At the present rate of price reductions on hard disks and the possibility of other methods of storing large amounts of data, this might well be all of us.

(Further Information: Amstrad Manual, page 282.)

SECTION 59
The shape of the tree

One problem about tree directories is the name – because the diagram that everyone shows you



looks more like the roots of a tree than its branches, unless you are in the habit of looking at trees upside-down. An alternative way of thinking about it is as a *family* tree, which starts with one ancestor and spreads out to all of the descendants.

However you like to imagine it, the diagram indicates that when you call for a directory display, all that you will get initially is the set of names that appear in the first line, called the root names. In the example, three of these filenames are TEXT, SPREADS and DATABASE, names that are chosen to indicate what type of files will occupy the three subdirectories. Note that these names are *not* names of files but names of other directories.

I have picked these names as illustrations only, because they show particularly well *why* it is useful to subdivide a directory in this way. Ironically enough, it isn't always possible to put some programs into a subdirectory and use them effectively. Some of the older word-processing and spreadsheet programs must be placed in the root directory along with their data files if you want to be able to use them directly.

■ SECTION 59
The shape of the tree

This is an important point if you have just bought a hard disk machine and want to add software, and it's equally important if you are buying software for a floppy-disk machine that you later intend to upgrade to hard disk operation. Only programs that can run under the MS-DOS sub-directory structure should be bought unless you have an overwhelming need to use a program regardless of its flaws in this respect. If you make use of only one type of program on your hard disk, then it hardly matters that you are confined to the root directory.

(Further Information: Amstrad Manual, pages 282, 283.)

SECTION 60
Trees in use

Since the GEM DESKTOP Master disk, Disk 3 (Green), is organised in tree form, you can use it as a demonstrator. Make a further copy, using DISKCOPY, and work with this copy only in this and the following sections. Do not experiment with your original Master disk, or even with a copy that you use daily.

When you print a directory of a disk that has a tree structure, only the files in the root directory will appear, along with the names for the other directories. The fact that there are other files in the branches (called the subdirectories) is indicated only by the word <DIR> appearing alongside the name for a subdirectory. In the example, taken from the Disk 3 directory, you can see that BASIC2, GEMAPPS, GEMDESK and GEMSYS are of this type.

```
 Volume in drive B is 46003
 Directory of  B:\

BASIC2          <DIR>       9-07-86   13:54
GEMAPPS         <DIR>       9-07-86   13:54
GEMDESK         <DIR>       9-07-86   13:54
GEMSYS          <DIR>       9-07-86   13:55
DOODLE   APP     28672       1-01-80    3:23
DISKCOPY COM      8832      20-06-86   16:57
NVR      EXE      8596       4-07-86    9:28
RPED     EXE      4612      14-07-86   15:37
DOODLE   RSC      2388       9-07-86   10:37
PTR                 3      10-12-86   16:21
       10 File(s)      11264 bytes free
```

Suppose, then, that you have the type of directory structure that is present in this disk, and you want to look at the items that are held in the BASIC2 subdirectory. You do this by using the command CHDIR \BASIC2, or the shorter version CD \BASIC2. This by itself does nothing visible, but the next use of DIR will produce the display showing the directory with the names BASIC2.APP, BASIC2.RSC and DEMO.BAS.

```
B>

 Volume in drive B is 46003
 Directory of  B:\BASIC2

.               <DIR>       9-07-86   13:54
..              <DIR>       9-07-86   13:54
EXAMPLES        <DIR>       9-07-86   13:55
PROGRAMS        <DIR>       9-07-86   13:55
BASIC2   APP     81920      11-07-86   20:15
BASIC2   RSC     11546      11-07-86   19:52
DEMO     BAS      6277      11-07-86   23:20
FIG2-1   BAS        81       8-12-86   17:18
        8 File(s)      11264 bytes free
```

SECTION 60
Trees in use

There are also entries that are marked EXAMPLES <DIR> and PROGRAMS <DIR>, meaning another layer of subdirectory. In addition, though, there are entries marked with a single dot and a double dot. The single dot represents the directory that you are currently in, which is the BASIC2 directory, and the double dot represents the one higher up the branch, the directory you have just come from, in this example the root directory. You can use the commands DIR. or DIR.. to display these.

Once again, the files in the next subdirectory down can be displayed by using, for example, CD EXAMPLES, followed by another DIR listing. When these subdirectories are displayed, the second line of the listing gives the directory path. In this example, it is B:\BASIC2\EXAMPLES, showing that we have moved from the root directory in drive B: to BASIC2, and from there to EXAMPLES.

```
B>

 Volume in drive B is 46003
 Directory of  B:\BASIC2\EXAMPLES

.            <DIR>       9-07-86  13:55
..           <DIR>       9-07-86  13:55
EXAMPLE3 BAS      5956   5-07-86  21:54
EXAMPLE1 BAS      1294   5-07-86  22:08
EXAMPLE2 BAS      2776  21-06-86   3:43
        5 File(s)      11264 bytes free
```

SECTION 61
Branching out

Working with tree directories in this way involves knowing how the tree is constructed, and in this respect, GEM uses simpler methods than MS-DOS. In this book, however, we will stick with the MS-DOS commands.

The first point is how you can tell what the structure of a directory is, assuming that you haven't drawn up a plan like that of the example at the time when you started to put data on to the disk. In most cases, you will probably have created the structure in a rather haphazard way, and will eventually start to wonder just what exists on the disk.

For that eventuality, MS-DOS has a command **TREE**, which you will normally use in the form TREE C: (assuming that the hard disk is drive C). It's a good idea to make the resulting listing appear on the printer with ample paper ready, because it can take a lot of space. This is because a TREE listing goes laboriously through each root directory name, listing its subdirectories in turn, rather than supplying a diagram.

Before you can use TREE, you should return to the root directory, by using the command CD\. You need the MS-DOS disk in drive A to use the TREE command, so that this will then be of the form TREE B: or TREE C:.

The sort of outpath that you get is illustrated opposite for Disk 3, and you will see the word **PATH** occurring frequently, along with the use of the backslash sign \.

SECTION 61
Branching out

```
B>
DIRECTORY PATH LISTING FOR VOLUME 46003

Path: B:\BASIC2

Sub-directories:  EXAMPLES
                  PROGRAMS


Path: B:\BASIC2\EXAMPLES

Sub-directories:  None


Path: B:\BASIC2\PROGRAMS

Sub-directories:  None


Path: B:\GEMAPPS

Sub-directories:  None


Path: B:\GEMDESK

Sub-directories:  None


Path: B:\GEMSYS

Sub-directories:  None
```

The word PATH is also a command in its own right, used to indicate how a tree directory is to be searched, and the backslash is used both as a divider between subdirectory names and as an abbreviation for the root directory. We'll look at these points in more detail later.

When you have made use of the TREE command for your hard disk, you should be able to draw up a tree structure showing how the directories are related. After some time, the number of directories on the disk will be more or less fixed, and all you will do from then on is to add, delete, or rename files in these various directories. When you get to that stage, the diagram for the directory structure is an essential part of your documentation. You should also print, at intervals, lists of the files in each subdirectory.

(Further Information: Amstrad Manual, pages 320, 247.)

■ SECTION 62
Working with branches

When you start work with a hard disk, you will probably start by creating a few entries into the root directory, and then probably into the first layer of subdirectories. The computer must, however, enter these names as subdirectory names, not as filenames, and to do this requires the use of **MAKDIR**, abbreviated to **MD**.

If, for example, you are in the root directory, and you want to create a sub-directory called TEXT, you do so by typing MD TEXT and pressing the RETURN key. This will cause some action in the disk drive, and when you use DIR from now on you will find the entry:

TEXT <DIR>

to indicate that TEXT is a subdirectory name, not a filename.

When you use DIR\TEXT to find if there is anything in this directory, you will get the following display:

```
A>dir\text

 Volume in drive A is DEMO
 Directory of  A:\TEXT

.            <DIR>      11-12-86   9:06
..           <DIR>      11-12-86   9:06
       2 File(s)      62464 bytes free
```

This shows that there are no files stored in this subdirectory, and the only entries are the dot and double-dot to indicate current and root directory. Note that the message '2 File(s)' is automatically delivered because there have been two lines printed – the MS-DOS DIR command does not distinguish between valid file names and the <DIR> or dot displays.

(Further Information: Amstrad Manual, page 309.)

■ SECTION 63
Working with subdirectories

You can create other subdirectory names of SPREADS and DATABASE in the same way, using the name that you have chosen following the MD command. This applies also to other subdirectories.

If you switch to the TEXT subdirectory, for example, you can create the further subdirectories of NEWTEXT, OLDEXT and BITS. This is done in the following stages, assuming that you are starting in the root directory. If you are not in the root directory, type CD\ (RETURN).

1 **Change to the selected subdirectory. To move to the TEXT subdirectory (assuming that you have used this name), type CD\TEXT.**

2 **Now use MD NEWTEXT to create a subdirectory of this name. You DO not need to use the backslash in this case, because you are starting in the correct subdirectory.**

3 **In the same way, create the other two suggested subdirectories, OLDTEXT and BITS.**

4 **The result of DIR should now appear as shown. The new directory structure is complete.**

```
A>md\spreads

A>

 Volume in drive A is DEMO
 Directory of  A:\TEXT

.            <DIR>      11-12-86   9:06
..           <DIR>      11-12-86   9:06
NEWTEXT      <DIR>      11-12-86   9:22
OLDTEXT      <DIR>      11-12-86   9:22
BITS         <DIR>      11-12-86   9:23
        5 File(s)     57344 bytes free
```

■ SECTION 64
Branching around

Note that you have to be in the TEXT directory in order to create these subdirectories of TEXT. If you are in the root directory when you use MD NEWTEXT , for example, you will have created a subdirectory called NEWTEXT all right, but it will be a subdirectory of the root, at the same level as TEXT rather than as a subdirectory of TEXT. The next thing is to find how to get from one branch to another.

You can change subdirectory by using the **CHDIR** command, abbreviated to CD. If, for example, you are in the root directory, then you can change to the TEXT subdirectory by typing CD\OLDTEXT. The use of CD with the slash and the name is enough to change from the root to any of the subdirectories that stem from the root.

The *path* is the route from the current directory (the display that you get using DIR) to whichever subdirectory you want to get to. The most straightforward path is from the root to any branch that stems from that root.

You do *not* use the backslash symbol if you are starting from a branch along a path, only when you start from the root, or if you need to go back to the root.

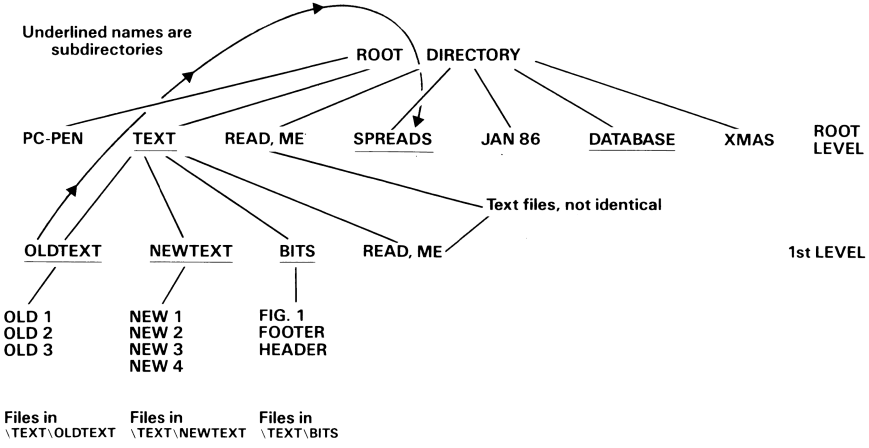
For example, if you are in the subdirectory TEXT and you want to find OLDTEXT, you need only type:

CD OLDTEXT

though you would need to use CD\OLDTEXT if you had been in the root directory.

You *must* use the backslash if you are going along any path that includes the root. As the illustration opposite shows, the path from OLDTEXT to SPREADS requires moving by way of the root directory, so that it needs CD\SPREADS.

■ SECTION 64
Branching around



The search for a subdirectory covers only the *one layer* below the directory you are presently using. It is always much easier to return to the root directory from any subdirectory, because this requires only the command CD\, using the backslash to mean 'root directory'.

(Further Information: Amstrad Manual, page 247.)

SECTION 65
Pathways

Having seen that we can create new subdirectories and transfer to them, using MD and CD , the next step is how to make use of the files that are located in the various subdirectories.

The easiest way to do this is to make some files, using RPED, and rename them so as to fit into the required directories OLDTEXT, NEWTEXT and BITS. This is done as follows:

1 **Return to the root directory, if you have not already done so, and load in RPED from another disk, or copy it on to the current disk and use it.**

2 **Use RPED to create three files. I have called these NEWONE, OLDONE and BITONE, as illustrated below:**

```
A>This is another text file called newone.
It is being used as a demonstration of
directory paths.

This is a simple text file called oldone
It will be used to demonstrate the use of
subdirectory commands.

A>
This is the text file called bitone.
It is being used like the others to
demonstrate directory paths.
```

3 **Copy the files to the subdirectory that you need, using the correct path in the destination name. Note that you *cannot* use RENAME unless the new name is to be in the same subdirectory. In this example, type:**

COPY oldone \text\oldtext\oldone.

4 **Copy the files NEWONE and BITONE in the same way, using the same path with the filename at the end.**

5 **Note that the machine takes the name at the end of the path as being the filename *only* if this name has not been used as a subdirectory name.**

The directories now look as shown below:

```
A>

 Volume in drive A is DEMO
 Directory of  A:\TEXT\NEWTEXT

.             <DIR>      11-12-86   9:22
..            <DIR>      11-12-86   9:22
NEWONE             101   11-12-86   9:43
        3 File(s)      52224 bytes free

A>

 Volume in drive A is DEMO
 Directory of  A:\TEXT\OLDTEXT

.             <DIR>      11-12-86   9:22
..            <DIR>      11-12-86   9:22
OLDONE             110   11-12-86   9:42
        3 File(s)      51200 bytes free

A>

 Volume in drive A is DEMO
 Directory of  A:\TEXT\BITS

.             <DIR>      11-12-86   9:23
..            <DIR>      11-12-86   9:23
BITONE             106   11-12-86   9:43
        3 File(s)      51200 bytes free
```

■ SECTION 66
Typing the files

To type any of these files presents no problem if you are in the correct subdirectory. The display of the subdirectories above was obtained by using a full path name, and this also needs to be used for commands such as TYPE, or anything else that requires a filename.

For example, to type the NEWONE file, you need to specify its path. To avoid confusion, delete the old names first, as follows:

1 **Return to the root directory by using CD\.**

2 **Type DEL OLDONE (RETURN), then DEL NEWONE (RETURN) and DEL BITONE (RETURN) so as to remove the original files.**

3 **Check by using DIR that the root directory no longer contains these files.**

4 **Place the OLDTEXT file on the screen by using:**

TYPE TEXT\OLDTEXT\OLDONE (RETURN)

5 **Repeat for the other files.**

6 **Note that when a full path like this is specified the starting point is the root directory and you always return to this directory after the command has been carried out.**

7 **You could, as an alternative, have used CD\TEXT, and then used commands such as TYPE OLDTEXT\OLDONE. After each such command, you would then return to the TEXT subdirectory.**

SECTION 67
Paths in action

Suppose, for example, that you are using the root directory and want to load into the computer a word-processor program called SHRDLU (anything that I can think of with WORD in it seems to be copyright), and that this file is in the TEXT subdirectory along with NEWTEXT, OLDTEXT and BITS.

The normal procedure for loading and running would be to type the name SHRDLU and press RETURN, and this would be sufficient *if* the program happened to be in the root directory. If the program is, as in this example, in the TEXT subdirectory, then an extended filename is needed.

This filename will show the pathway to the file, just as was needed in the CD command, so that it becomes \TEXT\SHRDLU, with the first backslash sign indicating that we start at the root, and the second used as a separator. If SHRDLU had been in the NEWTEXT subdirectory, you would then have needed the name \TEXT\NEWTEXT\SHRDLU to load and run the program.

This enhancement of the filename to include the pathway through the directory is used also by other commands. The commands DIR, COPY, DEL, RENAME and many others can all make use of this type of extended name, allowing you to carry out actions on files that are not in the directory that you are currently using. Note, however, that RENAME allows the use of a path only in the *old* filename, and the new name will be placed in the *same* subdirectory. Use COPY if you want to place a file from one subdirectory into another.

There is, however, a restriction on the extent to which you can do this path specification. No path instruction from the root directory to the lowest subdirectory can use more than 63 characters, so that the number of levels is limited to eight.

The name length for a subdirectory is limited to a maximum of is 8 characters, and longer names will be chopped down to this limit. Most users, however, will never need to use eight levels of subdirectories, even with a hard disk, and certainly not with floppies.

(Further Information: Amstrad Manual, page 284.)

■ SECTION 68
Sharing program files

Suppose, keeping to the example that we have been using up until now, that the subdirectory NEWTEXT contains a number of word-processor files, the chapter files for a book that is being written.

Now in this level of subdirectory, there is no spreadsheet program, and you might expect that you would have to change to the appropriate directory in order to use a spreadsheet. This is not the case, however, and one *program* file can be used from any directory by specifying the path to the file, using the PATH command.

Note that this means *program* files, not data files. You can in this way gain access only to program files, mainly of the .EXE or .COM kind, which you would normally activate by typing the name alone. You cannot use a PATH command to gain automatic searching for text files, for example, but you *can* use another command, **APPEND**, for this purpose(see Section 69).

Whatever you specify using PATH, the program that you are looking for will always be searched for in the current subdirectory. If you have used a PATH command previously, however, the other directories specified in the PATH will also be searched, and the PATH command would *always* start with a backslash to specify starting from the root.

In this particular example, by using PATH \SPREADS you will have specified that the search for the file you want, perhaps a spreadsheet called CROSSEYE, will start with the current subdirectory, then the root directory and then the SPREADS subdirectory. If you are in any other subdirectory, then, typing CROSSEYE will result in the program of that name becoming available. Any files that it creates will be placed in the subdirectory that you are currently using.

You can therefore use PATH to make several programs available without the need to keep copies in each subdirectory. To check what path you have used, type PATH (RETURN). To close down all paths, type PATH; . To specify more than one path, separate the paths by semicolons, for example:

```
PATH \SPREADS;\TEXT\OLDTEXT \DATABASE
```

(Further Information: Amstrad Manual, page 259.)

■ SECTION 69
Other ways

Suppose that you have a word-processor program in the root directory, or in a subdirectory. You might then want to be able to get at text files in any of your other subdirectories. The command that allows this is APPEND .

For example, using:

APPEND \TEXT\OLDTEXT;\TEXT\NEWTEXT;\TEXT\BITS

would provide paths to all of the files in OLDTEXT, NEWTEXT and BITS, from wherever you wanted to use your word processor. Since the action of APPEND is so very similar to that of PATH, we'll spend no more time on it here.

There are also a few commands relating to directory structures that you should not try to use until you have very much more experience. They are listed here briefly only for the sake of knowing what they do. It's unlikely that you will want to use these commands until you have more experience in any case.

The **JOIN** command makes a whole drive useable as a subdirectory of a root on another drive. In other words, if you have a hard disk, you can create a subdirectory which will allow access to drive A, but without needing to specify drive A. This allows you to use anything in drive A as if it were part of a subdirectory of drive C.

SUBST allows you to use a drive letter E to Z (*not* A to D) to represent some path to a subdirectory. You could, for example, assign drive X to the path C:\FOREIGN\ORDERS\UNPAID, so that to make use of files in the UNPAID subdirectory, you need only specify X:filename.

XCOPY works like COPY, and allows you to copy a file from one subdirectory to another. The command allows the use of several modifiers, and will create a subdirectory name if needed as the destination for the file(s).

(Further Information: Amstrad Manual, pages 244, 256, 264, 325.)

SECTION 70
Clearing up

The most difficult part of using a directory tree consists of closing down a directory. A root directory cannot be deleted, and you cannot delete the subdirectory that you are currently using, so it's always desirable to start any deletion operations from the root directory.

If, taking the usual example, you wanted to close the OLDTEXT subdirectory, you would first have to DEL each file in this directory, using DEL TEXT\OLDTEXT\BOOK*.TXT to show the path to the files (assuming that the files are called BOOK1.TXT, BOOK2.TXT, and so on), and using the wildcard so that each file is deleted, assuming that each file starts with the word BOOK. You could, of course, use *.* to represent any file in this subdirectory.

Only when each file has been deleted from the subdirectory can the subdirectory itself be deleted, using RD OLDTEXT.RD is short for RMDIR, and it cannot, remember, be applied to the directory you are presently using. In other words, you cannot remove subdirectory OLDTEXT if you are working in this directory, you must work either from the root directory or from another subdirectory.

The requirement to empty the directory of all files before deleting it can sometimes be a nuisance, though it does offer a considerable safeguard against deleting valuable files by making you consider all the files that have to be deleted. You can, however, short-circuit all this work if you are using GEM.

A subdirectory in GEM is represented as a folder. To delete the subdirectory, you use the mouse to point into the folder and then drag the folder to the bin. This can be done without any knowledge of what is contained in the subdirectory, so you need to be very sure that you have selected the correct folder. I loathe and detest symbols, and I would want to be very certain that I was deleting the correct subdirectory when I used this method.

(Further Information: Amstrad Manual, pages 318, 149.)

PART FIVE

Peripheral control

SECTION 71
Printers

Whenever your use of a computer starts to involve anything that needs to be kept on paper, you need a printer. A printer is virtually essential with any computer that uses a disk system, because the use of a disk system is in itself almost a guarantee of serious use.

The reasons for using a printer are obvious if you use the machine for even the smallest business purposes. You can hardly expect your accountants or your income-tax inspector to look at accounts that can be shown only on the screen. It would be a total waste of time if you kept your stock records with a computer, and then had to write down each change on a piece of paper, copying everything from the display on the screen.

For all of these purposes, and particularly for word processing, the printer is an essential part of the computer system. Output on paper is referred to as **hard copy**, and this hard copy is essential if the computer is be of any use in business applications. For word-processing uses, it's not enough just to have a printer, you need a printer with a high-quality output with characters as clear as those of a first-class electric typewriter.

Even if your computer is never used for any kind of business purpose, however, you can run up against the need for a printer. If you use, modify or write programs, the printer can pay for itself in terms of your time. Trying to trace what a program does from a listing that you can see only a few lines at a time on the screen is totally frustrating.

Quite apart from anything else, the BASIC 2 of the PC relies a lot on the use of GEM and its windows, and it can often be very frustrating trying to check through a program on the screen.

■ SECTION 72
Printer types

Granted, then, that the use of a printer is a high priority for the really serious computer user, what sort of printers are available? The answer is any type that comes with either a Centronics parallel interface or the (less used) 'serial interface'. This accounts for all but a handful of special-purpose printers that are designed to be used only with some particular computers.

You have the usual choice of the different printer mechanisms. Printers that are used with small computers will use one of the mechanisms that are listed below:

Dot matrix
impact

Type impact
type stalk
daisywheel
thimble
type-band

Plotters
graphics printers
X-Y plotters

Ink jet
single colour
multicolour

Laser
laser fast printer

Of these, the impact dot-matrix type is the most common. A dot-matrix printer creates each character out of a set of dots, and when you look at the print closely, you can see the dot structure. Most of the dot-matrix printers are impact types. This means what it says, that the paper is marked by the impact of a needle on an inked ribbon which hits the paper. Dot-matrix printers made by Epson, or compatible with the Epson types, can reproduce graphics and other screen patterns from the PC machines. Non-impact types of dot-matrix printer that use specially sensitised paper have no place nowadays in this list.

The ultimate in low-cost print quality at the moment is provided by the daisy-wheel printer. This uses a typewriter approach, with the letters and other characters placed on stalks round a wheel. The principle is that the wheel spins to get the letter that you want at the top, and

then a small hammer hits the back of the letter, pressing it against the ribbon and on to the paper. Because this is exactly the same way as a typewriter produces text, the quality of print is very high. It's also possible now to buy a combination of typewriter and daisy-wheel printer. This looks like a typewriter, with a normal typewriter keyboard, but has an interface connection for a computer. You can use it as a typewriter, and then connect it to the computer and use it as a printer. Machines of this sort are made by leading typewriter manufacturers such as Silver Reed, Brother, Triumph-Adler, Smith-Corona, and others. If you need a typewriter as well as a printer, then this type of machine is an obvious choice.

If you want first-class typesetting quality, with a view to *desktop publishing*, then you need a laser printer. The principles involved are beyond the scope of this book.

(Further Information: Amstrad Manual, page 12.)

SECTION 73
Interfaces

The printer has to be connected by a cable to the computer, so that signals can be passed in each direction. The computer will pass to the printer the signals that make the printer produce characters on the paper, but the printer must also be able to pass signals to the computer.

This is because the printer operates much more slowly than the computer. Unless the printer contains a large memory *buffer*, so that it can store all the signals from the computer and then get to work on them at its own pace, some sort of *handshaking* is needed. This means that the printer will accept as many characters as its memory will take, and then sends out a signal to the computer which makes the computer hang up.

When the printer has completed a number of characters, (one line, one thousand, or possibly just one character), it changes the handshake signal, and the computer sends another batch. This continues until all of the text has been printed. This *can* mean that you don't have the use of the computer until the printer has finished, but the PC provides for the use of *background printing*, meaning that printing is done by taking characters from a file stored on a disk, and this is done while the machine is not otherwise engaged, even when you are using it for other purposes. This is possible because the computer needs only a very short interval between commands to send another batch of characters to the printer from a disk.

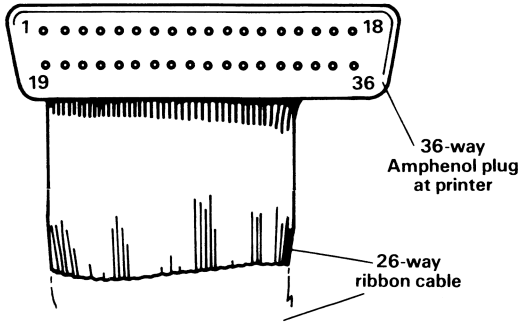
Printers can be very slow, particularly daisy-wheel types. Even the fastest dot-matrix printers can make you wait for a minute or more for a close-typed page. Many programs such as WordStar incorporate their own buffer arrangements so as to let you get on with other tasks while printing is done in the background.

■ SECTION 74
Parallel interfaces

A parallel interface uses a cable with many strands, usually 20 at least, to connect the printer to the computer. Each character is sent to the printer as a set of eight electrical signals, so that eight strands of the cable are used for this purpose, and the rest are used for synchronising the signals.

A parallel interface is easy to use, mainly because the standard that was set by the manufacturer, Centronics, has been adopted by so many other manufacturers. If you have a printer that uses the Centronics parallel interface, then all that you will need to connect to your PC is a suitable cable.

Parallel cables are usually short, about a metre long, and are fitted with a different type of plug at each end. The plug at the PC end is a 25-pin D-type plug, of the type that most machines use for a serial printer. The plug at the printer is the standard 36-pin type of Centronics plug.



Longer cables for parallel printers can be bought, as can extenders, but on long cables there is more likelihood of interference between the lines, causing some misprinting. Two metres should be taken as a maximum length. Once the cable is fitted, all of the normal printing actions can be used.

The output of a command such as TYPE, for example, can be sent also to the printer by pressing CTRL-P before pressing RETURN on the TYPE command. The contents of a screen can also be printed by pressing the PrtSc key (SHIFT-*) at any time. There is also an MS-DOS command PRINT which is followed by a filename and which will print out the contents of a file. This command will be dealt with in more detail later.

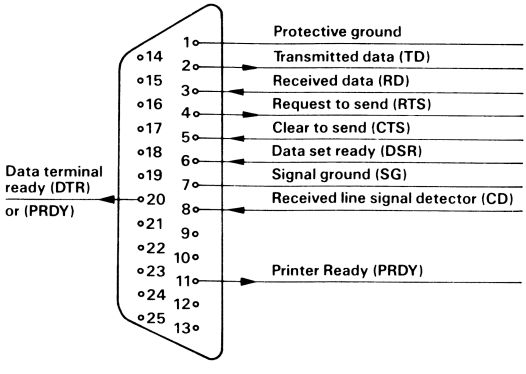
(Further Information: Amstrad Manual, page 497.)

■ SECTION 75
Serial interfaces

A serial interface deals very differently with the eight signals that are needed for each character. As the name suggests, the signals are sent out one at a time rather than together. This means that at least seven, and often eight, signals have to be sent for each character, and in practice this number must be ten or eleven, to allow for 'start and stop' signals which are used to mark where the signals for each character start and stop.

This system uses less cabling, because only two strands need to be used for signals, and the cables can be longer, because there's no risk of one signal interfering with another when there is only one being sent at a time.

The standard system is called **RS232**, and you can easily buy a cable that will connect your PC to any RS232 printer, or to any other device that uses the RS232 connection. The trouble is that standards in such cables are widely ignored.



If you want to connect your PC to a serial printer, you need a serial printer cable, often called a non-modem cable. A modem cable is used to link the computer to a device (modem) which will allow computers to be linked over the telephone lines, and the connections for the two cables are not the same, though they are both termed RS232.

You must then arrange for printing to take place by arranging for the computer to send out serial signals that are exactly matched to the signals that the printer expects. This is called *setting protocols*.

■ SECTION 75
Serial interfaces

These protocols include the number of signals that will be transmitted per second (baud rate), the number of signals for each character (usually 7 or 8), the number of signals used to indicate the end of a character (stop bits, usually 1 or 2), and parity, which can be None, Odd or Even. Parity is a way of checking for errors in the transmitted signals, and is seldom used for printing, so that this option is nearly always set to None

The standard settings of the computer for serial transmission depend on which operating system you are using. The use of DOS Plus selects the following:

Baud rate	9600
Parity	None
Data bits	8
Stop bits	1

These parameters are easily altered either for outputs or for inputs. When MS-DOS is used, the standards are:

Baud rate	must be set by MODE command
Parity	Even
Data bits	7
Stop bits	1 except for baud rate of 110, which uses 2.

(Further Information: Amstrad Manual, pages 348, 500.)

■ SECTION 76
Setting protocols

Practically all serial printers can be adjusted, usually by setting switches internally, to whatever protocols are required. If you can set your printer to a baud rate of 9600, no parity, 8 data bits and 1 stop bit, then it will match perfectly the settings used by default when under DOS Plus.

To make the MS-DOS serial output match these settings, you need to use the MODE command:

1 **Place the MS-DOS Master disk in drive A.**

2 **Type the command:**

MODE COM1:9600,N,8,1,P

and press (RETURN) to set the protocols. The letter P is used when the output is to a printer.

3 **Type the command:**

MODE LPT1: = COM1: (RETURN)

which sends printer signals to the serial output. By default, all printer signals are sent to the parallel output, coded as LPT1:. By using this command, the signals are sent to the serial port instead, so that all printer actions involving the use of CTRL-P, the ScnPrt key, or PRINT, will make use of the serial printer.

4 **If you need to return to the use of a parallel printer, use MODE LPT1: (RETURN).**

Note that the MODE command is also used for a large variety of other purposes, including some actions on parallel printers.

(Further Information: Amstard Manual, page 348.)

■ SECTION 77
Line feed and carriage return

A problem that you are bound to run up against when you use any printer is that of line feed and carriage return. A lot of computers send out only one code number, the carriage return code (13) at the end of a line. Other machines, including the PC, send both the line feed (code 10) and carriage return codes.

Printers are arranged, therefore, so that either possibility can be catered for by a switch. If you connect your printer and find that every line is double-spaced, then this switch will have to be set to the opposite position. When printers are used along with word processors, it's easy to arrange for the special commands to be sent to the printer. These are the commands that cause bold type, underlining, margin setting and so on. No such arrangements are possible in any simple way when using MS-DOS, but a method of sending such commands to your printer is noted in Part 8.

■ SECTION 78
Other modes

MODE is a command that can carry out a large number of very different actions, and the only thing that these actions have in common is that they concern peripherals like printers, the screen, modems and so on. The letters that follow the MODE command decide which peripheral will be affected, and the codes that are used are listed below:

LPT1: Parallel printer
COM1: Serial output
40, 80, CO40 or CO80: Screen

Of the four options that MODE can take, the use in controlling the printer actions is probably the most useful for the 1512 owner, and this is the mode that we shall pay most attention to.

We have already dealt with the use of MODE to set up for a serial printer, but it has applications to a parallel printer. In order to make the MODE command refer to the parallel printer, you need to use the code LPT1: following the MODE command word, separated by a space.

You can then specify a few features of the printer output by using numbers that have to be placed in the correct order. The first number following LPT1: will determine the number of characters per line of print, with the choice of 80 or 132. When you use a word-processor program, of course, this will separately be determined by the program, and you should not attempt to change the printer settings by the use of MODE.

In addition, there is no guarantee that the printer will take any notice of the code that is sent to it for some actions. The number of characters per line on a daisywheel is set by the type of daisywheel that is being used and by the switch settings on the printer, and the same is usually true of many dot-matrix printers. None of my printers responded to the MODE LPT1:132 command.

The following figure in the MODE LPT1: command sets the spacing between lines at 1/8 or 1/6 inch. The default is 1/6, but using MODE LPT1:,8 will set to the closer spacing. This certainly works on the Epson printer, but might not have any effect on others.

SECTION 78
Other modes

MODE LPT1:,6

```
A>
This is a simple text file intended to demonstrate some of the printer
set-up actions. The file has been created using RPED and is stored
under the filename of TESTIT.
```

MODE LPT1:,8

```
A>
This is a simple text file intended to demonstrate some of the printer
set-up actions. The file has been created using RPED and is stored
under the filename of TESTIT.
```

Note that if the first figure is omitted, as above, the comma must be retained in the MODE command. Finally, you can use the letter P at the end of the numbers, as suggested by the manual, but it has no noticable effect on my printer action.

(Further Information: Amstrad Manual, page 348.)

■ SECTION 79
Screen modes

Another use for MODE is in determining screen character sizes. You have the choice of 40 or 80 characters per screen width, and the default is 80. By using MODE 40 , you can switch to 40 characters per line, giving much larger displays in which a directory listing fills the width of the screen. You can return to normal by using MODE 80. Avoid the use of BW40 and BW80 on the Amstrad machine unless you are certain that the monochrome adaptor has been fitted. This is not necessarily the case just because the machine uses a monochrome monitor.

(Further Information: Amstrad Manual, page 348.)

■ SECTION 80
The PRINT command

PRINT is a command whose effect is to print an ASCII file of text on the printer. In this respect, it might appear to be rather like the use of TYPE with the CTRL and P keys used to echo the screen printout to the printer, but PRINT is rather more subtle than that. When you use PRINT, followed by the name of a file, the file will be printed on paper, but the computer can still be used for other tasks. PRINT is a time-sharing action, one that is done by loading in text in batches from the disk, storing the text in memory, and sending the text to the printer in any spare time that the computer happens to have.

You cannot, of course, remove the disk from which the computer is reading the text, but you can load in a program from another drive or run such a program while the printing is being carried out. You cannot, however, carry out any other printing action until the printer is free for use again. The action of PRINT can set up a queue of files to be printed, and unless you terminate the action in some way, you cannot use the printer for any other printing actions.

(Further Information: Amstrad Manual, page 311.)

■ SECTION 81
Setting up PRINT

When you make use of the PRINT command for the first time in a computing session, you might want to change some of the settings. These settings can be changed by issuing command letters in the first PRINT command, but at the first use of PRINT, you can also make one alteration without any special effort.

This option is of the name of the device to which print characters will be sent, and the option takes the form of the question: Name of list device [PRN]. You then press RETURN unless you are using a serial printer, in which case you can type COM1: and then press RETURN.

There are six other options that can also be chosen at the time of the first use of PRINT. This can be done at the same time as you specify a list of files to be printed, or you can type PRINT by itself followed by these selections, as follows:

- **/B:1024** sets the print buffer store to 1024 characters. The larger the number here, the easier it is to interleave the printing action with any other computing. Use numbers that are multiples of 512 preferably. The default is 512.
- **/Q:12** allows you to use a queue of up to 12 files waiting to be printed. The default is 10.
- **/S:5** means that the computer will spend one sixth of its time in printing, using a ratio of computing to printing of 5:1. The default is 8. It is not always possible to predict what effect this will have on the rate of printing, because if the printer has itself a large buffer memory, the effect may be negligible.
- **/U:1** and **/M:2** are default values that are best left alone unless you want to experiment. The numbers that can be used can range from 1 to 255, and they control the timing of switching over from computing to printing and from printing to computing.

(Further Information: Amstrad Manual, page 311.)

■ SECTION 82
Practical PRINT

The use of paper with PRINT is not quite like the use when the TYPE with CTRL-P is used. When you use TYPE with CTRL-P, the printing stops at the same time as the listing, but at the end of a PRINT document, the printer takes a new page so as to separate the documents that are being printed.

In addition, PRINT allows the use of a queue of files. For example, using:

```
PRINT TXT1 TXT2 TXT3 TXT4
```

would cause all four of these files to be printed, with a blank sheet of paper separating each pair of documents.

You can add more files to the queue while PRINT is operating by using **/P**. For example, by typing PRINT TXT5/P , you add the file TXT5 to the existing queue, but only if there is space for it. With the default value of 10 files in the queue, this is seldom a problem.

You can remove files from the queue by using **/C** or **/T**. The use of /C removes a file, so that if you type:

```
PRINT TXT4/C
```

before this file has been printed, it will be removed from the queue. Using PRINT /T removes all files from the queue.

You can use wildcards in the filename if you want to place a number of files in the queue with a short command. You can also use paths in the filenames. Be careful about queue lengths, because this, and the other settings that are made when PRINT is first used, cannot be changed until the machine is reset and restarted with MS-DOS.

(Further Information: Amstrad Manual, page 311.)

■ SECTION 83
Graphics printing

Most applications programs like spreadsheets make provision for printing any graphics that can also be seen on the screen, using the PrtSc key. This is done by using the **GRAPHICS** command of MS-DOS.

If you have a pattern on the screen that is not printed by the programs that generated it, then you can use the command GRAPHICS, with the MS-DOS disk in Drive 1. Subsequently pressing the PrtSc key will print the pattern on to paper.

Note that this assumes that you have a suitable printer (such as an Epson), that it is switched on and connected, and that you can keep the pattern on the screen while you use the GRAPHICS command and PrtSc key. In general, you would set up by using the GRAPHICS command *before* you ran the program that generated the pattern.

(Further Information: Amstrad Manual, page 254.)

PART SIX

Checks, errors and error messages

■ SECTION 84
Message formats

When a command cannot for some reason be obeyed you can expect to get an error message appearing on the screen. Some early types of DOS were notorious for cryptic error messages, whose meaning was rather more difficult to discover than the actual error itself.

MS-DOS has improved on these types of message and, in general, it's now fairly easy to find from the nature of the message just what went wrong. What is more important, though, is to decide what to do about it. The more you know about the MS-DOS error messages, the easier it is to take the correct type of action to remedy each error as it occurs. First of all, though, we need to look at why errors should occur at all.

Some errors are unavoidably due to carelessness. If, for example, you put a disk in the machine and type the name of a program that does not exist on that disk, you will get the message

Bad command or filename

which makes the error clear, and should prompt you to use DIR to find just what is on the disk that you have used.

Another version of this type of error will occur if you start up the machine without using the system disk, or with no DOS files on the hard disk, if a hard disk is in use. For a floppy-disk machine, loaded with MS-DOS from drive A, the message that you get when you use a disk without the DOS files will be self-explanatory:

Wrong disk. Insert a SYSTEM disk and press any key

This is quite enough to advise you of what has happened and what has to be done. If, however, the disk with the MS-DOS files on it has become corrupted (you should not have placed it on top of the monitor!), then you may find that the loading of the DOS starts in the usual way, and then stops with the message:

Disk boot failure

to advise you that the loading (or booting) of the DOS files has been unsuccessful, so that the computer cannot be used.

The first thing to try when you get this message is to switch off and start all over again, using the same disk. If the same message is received, you should use another copy of the System disk or another disk with the DOS files on it.

It's at this point that you become very grateful that you made a backup copy when you took delivery of the computer. No matter how

urgently you need to make use of the machine, once you have experienced this error, you should make another copy of the system disk, or another copy of whatever disk you use to start the machine.

If you don't, it will quite certainly be forgotten, and you will find some day that you have a splendid collection of disks, none of which will allow you to start the machine.

■ SECTION 85
Errors in programs

The errors that we have looked at so far concern starting up the machine with the correct program loaded in. What happens when a program is running very much depends on how the program is written. Some programs are written in such a way that you are most unlikely to encounter any error message from the DOS. This is because anything that could cause a DOS error is checked, and the program issues an error message of its own, prompting you for a correction.

A DOS error will occur only for some really serious fault that makes it impossible for the program to continue. It's important to be able to recognise whether an error message comes from the program that you are running or, because of a more serious error, from the DOS.

Many types of error bring up the DOS message about the error, but also the message

Abort, Retry, Ignore?

This calls for you to enter one of these letters A, R or I, and press RETURN. If the error is one that can be sorted out by, for example, changing to another disk or removing a write-protect label, then you can do this and use the R reply.

This action (of which more later) allows the normal program work to continue provided that the problem has been sorted out.

■ SECTION 86
Other errors

Suppose you are using a word processor and you find an error message that comes from the word processor itself. You can then take whatever steps are needed (insert correct disk, remove write-protect label, delete some files on disk, and so on) and continue. If, however, you get a DOS error message and the DOS prompt symbol > , then you are in rather deeper trouble. Most programs use a different prompt *(not* A>) so that you can see when you have returned to DOS.

Once a program has returned *permanently* to DOS in this way, you cannot restart the program without losing all the data that it had collected. If, for example, you had typed some two thousand words into your word processor at the point when it returned to the DOS, then this text may be lost.

I say *may* because a lot depends on the type of word processor and your own mastery of DOS. Some word processors will automatically save text to the disk at intervals, so that if you had typed 2000 words, then probably some 1800 or so would have been saved, and could be recalled by restarting the program and loading the saved text in from the disk.

This type of automatic save action is very valuable, and the type of word processor that keeps all of its current text in memory can lead to nervous hysteria among authors who have suffered the experience of losing a large chunk of text. Even a comparatively small loss can still be serious, however, if it took a lot of effort to think up, and for such cases, more drastic measures are needed.

■ SECTION 87
Recovery

(This advanced topic should be omitted on a first reading.)

If your word processor returns to DOS and the > prompt, the text that you were working on will still be retained in the memory, provided that you don't switch off the computer or attempt to restart the word-processing program. Since the text is still in the memory, it can be saved on to a disk as a file, *if you know how.*

The snag is that this involves a much deeper knowledge of the workings of the machine and of a utility called **DEBUG**. Details of how to use DEBUG to recover a file of this type are beyond the scope of this book, but a brief outline of the DEBUG program itself is given in Part 8. If you want to investigate further, details are in advanced books.

Once the file has been saved by the use of DEBUG or other file-saving program, you can take the disk out, restart the computer with the word-processor, and read this file – then check the text and make a backup copy. The important point is that DEBUG requires you to specify location numbers, called *addresses* for the first and last characters in the file, and these numbers will be different. For a given program the file always starts in the same place, but the end point depends on how many characters are in the file.

When you replace the word processor by your spreadsheet, you will find that the file of data that this program keeps will start at a quite different place in the memory. This is because each program consists of a file of codes which are loaded into the memory, and the text or other data that the program then generates is simply placed in the rest of the memory, occupying places that have higher address numbers. Each different type of program that you use will therefore have a different starting address for its data, and only if you have a note of all these places can you use DEBUG to recover data. This recovery is possible only because DEBUG is a comparatively short program, so that it takes up less of the memory than the program that originally put the data into place.

Note, however, that using DEBUG is getting into the realms of advanced programming, and careless or unknowing use of DEBUG can lock up the computer or corrupt a disk. If you experiment with DEBUG, work only with backup disks, and be prepared to lose data from the memory. Consult a guide to MS-DOS programming techniques for more information on DEBUG. You may also find that magazines publish from time to time routines for the recovery of *lost* data from well known programs like WordStar.

■ SECTION 88
Errors involving utilities

Quite a number of the error messages of the DOS can appear while you are using the utilities of the System disk, such as FORMAT, DISKCOPY, RENAME and others. In general, these errors are less serious because they are unlikely to cause loss of data, and the messages are straightforward, indicating what should be done in each case.

The most common of all error messages is

Bad command or filename

and this is delivered usually because you have forgotten that the utility that you want to use is external. You cannot, for example, use CHKDSK on a disk in the drive you are using unless this disk contains the CHKDSK utility. The method of using such utilities is to use a command of the form CHKDSK B:, with the MS-DOS Master disk in drive A. This applies whether you have a single drive or twin drives.

Some utilities dealt with in this Part have not been mentioned earlier. They are included here because they are, in general, used to discover errors, so that their error messages are of considerable importance. The action of each utility and its normal message are described, followed by the types of error message.

Not all messages indicate errors. Some are simply information that may not be of much importance to anyone other than a programmer. As always, the most serious errors are those that indicate that a disk has a fault (track or sector unreadable or unusable), or that a file cannot be copied because of lack of space.

Since backing up and integrity of files are important, any disk with a faulty track or sector should be scrapped ruthlessly, providing that all readable files have been copied. There's no point in being sentimental about a disk that cost 80p, and if you leave it around, it's almost certain to be used at some time when you are in a hurry, resulting in loss of data that you thought was safely recorded.

Equally, if you are backing up a file and you find that the destination disk has insufficient space, you should at once repeat the backup with a fresh disk. Do you keep a supply of new, *formatted* disks? The important point is that if you don't have a formatted disk available, you can't carry out the backup, and if you have no other copy of the data, then you may be in difficulties.

►

■ SECTION 88
Errors involving utilities

It may be that you can delete some files from a disk, files that are either unwanted or are already adequately backed up, and that this will release enough space to allow the present file to be saved. The important point to realise is that the FORMAT command is external, not internal. Using FORMAT therefore wipes out the program that you are using, and that, or a data file from that program, may be what you want to save. Get into the habit of formatting disks as soon as you receive them, and marking the boxes 'FORMATTED' when this has been done.

■ SECTION 89
CHKDSK messages

CHKDSK, followed by drive letter, is used to check and, if possible, carry out some repairs on, a disk that may have been damaged, or simply to report on how much of a disk is free for use. Following the drive letter, you can use **/F** to fix any repairable damage, and/or **/W** to put progress reports on the screen.

CHKDSK can also be used with a filename (including directory path) following, so as to report on a file. The /F and /W extensions can be used also in this application.

The normal message shows total disk space, amount used in files, amount left; also total memory and amount used by MS-DOS.

- *Allocation error, size adjusted* means there are errors in the allocation of disk sectors. Use /F following filename in CHKDSK command to repair.
- *Convert lost chains to files* means that CHKDSK has found some data that has no filename, and can convert this into a file. Answer Y to convert, but only if the data is likely to be text. Answer N to delete.
- *Corrections will not be written to disk* indicates that repairable errors have been found, but not repaired. Use CHKDSK with /F following filename to correct errors.
- *Entry has a bad attribute or link or size* means that there is a fault in the directory structure.
- *Errors found, F parameter not specified.* Corrections will not be written to disk. Try using CHKDSK again, with /F following file name.
- *Has invalid cluster, file truncated.* A file has been wrongly stored, and the incorrect sectors have been deleted so as to shorten the file. If this was a program file, it is now useless.
- *Probable non-DOS disk.* Continue (Y/N)? You are probably using a disk from a different type of machine. Remove this disk and label it as suspect.

(Further Information: Amstrad Manual, page 330.)

■ SECTION 90
COMP messages

COMP is used to compare two files that ought to be identical. It's not a command you normally need to spend a lot of time with, because if two files that should be identical are not, then the remedy is to make new copies provided that you know which is the correct file. COMP offers a large number of options, of which the most useful are /**A** and /**C**.

Using /A stops the program from displaying each of a set of differences, and instead shows the last matching portion found, and the next one, using a set of dots to indicate where the differences were found. Using /C causes all characters to be treated as upper case, so that a comparison between 'DISK' and 'disk' will not show as an error.

The form of COMP is:

COMP /A/C A:OLDFILE.TXT B:NEWFILE.MSG

in which OLDFILE is the *standard*, correct file, and NEWFILE is the copy which is being checked. The full filenames and paths can be used.

The normal message is:

fc: no differences encountered

Error messages:

- *fc: cannot open filename* Incorrect specification of a file or drive.
- *fc: out of memory* Not enough memory to store the files – unlikely on the Amstrad PC unless you have reserved memory in some way, perhaps for RAM-disk.
- *usage: comp (message)* indicates that one of the options was incorrectly specified. Avoid using any options when you are comparing .EXE or .COM program files, unless you know what you are doing.

(Further Information: Amstrad Manual, page 296.)

■ SECTION 91
DISKCOMP messages

DISKCOMP is used to compare the contents of two disks. The normal form is DISKCOMP if you have one drive only, or DISKCOMP A: B:, if you have two drives, but the DISKCOMP program itself *must* be on the disk in drive A:.

The normal message is **Compare OK**. Options are **/1**, meaning compare only one side of each disk, and **/8** meaning compare only the first eight sectors of each track. You would not normally use these rather specialised options.

Error messages are:

- *Compare error on side x, track y* Shows where error exists. This is seldom really useful, and you should make another copy of the disk and then reformat the incorrect copy.
- *FIRST diskette bad or incompatible* The formatting of the first disk is incorrect. You need to run CHKDSK on this disk before proceeding.
- *SECOND diskette bad or incompatible* The formatting of the second disk is incorrect. Try CHKDSK on this one before attempting to use DISKCOMP again.

(Further Information: Amstrad Manual, page 332.)

■ SECTION 92
DISKCOPY messages

The use of **DISKCOPY** has already been dealt with. If the disk to which you are copying has not previously been formatted, you will get the message

Formatting while copying

and the copy action will take more time. This is not a fault.

The normal message at the end of copying is simply

Copy another diskette (Y/N)?

Error messages are:

- *Copy not complete* Some parts of the disk could not be copied. Some commercial disks are copy-protected and will cause this message; another possible cause is a corrupted disk being copied.
- *Disks must be the same size* The target disk has a different format, and should be reformatted.
- *Target diskette is write-protected* You should check why the disk is protected, and remove the write-protect tab if you really want to use this disk.

(Further Information: Amstrad Manual, page 334.)

SECTION 93
Device errors

The usual DOS error messages are concerned with a variety of errors, and they do not take any standardised form. One group of error messages, however, is concerned with device errors and is of a completely standardised form with a standardised response.

The devices that are concerned are mainly the printer and the disk drive, because these are the devices most likely to cause problems such as having no paper in the printer or no disk in the drive. The error message consists of the type of error, whether this occurred on reading or writing, and what device was responsible. The response can be to press one of three keys, A, R or I, meaning Abort, Retry, or Ignore. In most cases, you will be able to sort out the problem and choose the correct response. The golden rule if you don't understand the fault is to try R (Retry) first in the (rather small) chance that the error won't recur.

Obviously, if you have made some change (like putting paper into the printer or putting in the correct disk) this option should have the effect of carrying out the command correctly. If you have done nothing, however, pressing R is more a act of despair than anything else, like playing newspaper Bingo.

Given a second refusal after pressing R, the next option is A (Abort). This will stop whatever command or program originally brought up the error, and return you to DOS, awaiting a command. If you were using a utility when the error occurred, then this is often the most sensible course, because it allows you to start all over again, perhaps taking some sensible precautions this time.

You might, if you have been using a program that generated data, have lost data by doing this, and this has already been mentioned. At least if the data is still contained in the memory, there's a chance of recovering it by saving it to a file if you know what technique to use.

Finally, using I(Ignore) means that the program or command should ignore the error and proceed. This isn't always possible; in fact for many errors it is never possible, and if it *does* work, then it's most likely that any data will have been lost for good, impossible to recover in any way.

Once again, if the cause of the error lay in a direct command, there will be no data involved, and I may be a reasonable response. In general, however, the Ignore option is one to avoid unless you know by experience that it is useful in some specific case.

■ SECTION 94
Common errors

The error types are summarised in the Amstrad Manual (page 517), in alphabetical order. A more useful order is the order of likelihood, because the 'Bad command', 'Write protect', 'disk', 'No paper' and 'Not ready' messages are by far the most common cause of device errors.

For a 'Bad command' error, the appropriate response is to Abort, and then retype the command correctly, checking that the command name exists on the disk. For the 'Write protect' error, you should check that you really want to use this disk, and if so, remove the write-protect tab and re-insert the disk (or insert another formatted disk). You can then press Retry.

The 'Disk' message usually means that you have no disk in the drive that you are using (is it the one that you expected to be using?) or that the drive door lever is not shut. Once again, correcting the fault and pressing Retry is the appropriate response. The same is applicable to the 'No paper' message, so that you should load paper into your printer, and then use R to get the printing started. The 'Not ready' message can apply to a printer (not switched on, or off-line) or to a serial port, and it means that you have to prepare the device for use in some way before using the R key.

(Further Information: Amstrad Manual, page 517.)

■ SECTION 95
Running repairs

The **RECOVER** command can sometimes be useful if you find that a disk of text files has become unreadable. You can copy the RECOVER program on to the disk, or use the damaged disk in drive B, with the MS-DOS disk in drive A. To recover all files on the disk, type:

RECOVER B: (RETURN)

using whatever drive letter is applicable.

The action of RECOVER should allow you to read all the files, though some may have been shortened. If you use RECOVER on program files, any files that have been shortened will be unusable. In general, RECOVER does no more than the use of the /F option with CHKDSK.

Another program, **REPLACE**, is intended as a way of avoiding repairs rather of carrying them out. REPLACE is used when you keep two sets of disks, of which one is a working copy and the other is a backup. Using REPLACE then allows you to replace only these files that have been changed in a days work, rather than make a backup of all the files. It's particularly useful for backing up fresh work stored on a hard disk, for example.

Since this is not a command that you would use until you have considerable experience, the description in the manual will be sufficient when you have need of it.

(Further Information: Amstrad Manual, page 316)

PART SEVEN

Diving deeper

■ SECTION 96
Inputs and outputs

A lot of applications for the computer call for you to enter data at the keyboard, and to see some kind of output from the screen. The keyboard and the screen are the main input and output devices of the computer. As far as the main processing part of the computer is concerned, the keyboard and the screen are peripherals, remote devices, and the general name of **device** is used to mean anything of this kind that is not directly involved in the computing actions, but is used in the input or the output of data.

The standard input is input from the keyboard and the standard output is to the screen. These devices are so often used together that there is a name **console** which means a keyboard/screen unit with the keyboard providing input and the screen displaying output. What makes the name important is that the first three letters are used as an abbreviation in some DOS commands, as we shall see.

Now we don't always want input to come from the keyboard. You might, for example, want to use your computer to place text into a disk file, but receive the text from the telephone lines through a modem. This is an increasingly important action now that so many office jobs can be carried out at home with a telephone link to the office – providing that the cost of the telephone line is less than the cost of office space.

We therefore need to be able to redirect the input so as to treat the ASCII codes that are coming in from the modem through the serial port as if they were coming from the keyboard. Once the text has been put into a file, it can then be used, for example, by a word processor. This allows you to let the input proceed quite automatically, to be dealt with later (so using the telephone lines at off-peak hours, for example).

■ SECTION 97
COPY in redirection

The COPY utility can be used for transferring data in paths other than from one disk to another, or from one disk filename to another on the same disk. This is done by replacing one or more of the filenames in the command by *device names*,the standardised three-letter names for items such as the printer, the serial port, the screen and so on. The names that are used for the devices that are fitted as standard in the PC1512 are:

- **CON** The console, keyboard or screen.
- **AUX** or **COM1** The serial port, for use with modems, connecting to other computers, or to serial printers.
- **PRN** or **LPT1** The parallel printer output.

Note: do not use COM2, LPT2 or LPT3 unless you are certain that devices have been fitted that use these names.

By using one of these names in place of a filename in the COPY command, you can force data to be sent to a device or taken from a device. For example, you could make the machine send everything that you typed on the keyboard into a disk file called TYPIT.TXT by using the command:

```
COPY CON TYPIT.TXT
```

pressing RETURN, of course, to execute the command. From that point on, everything that you typed would be saved to disk until you typed the standard *end* signal of CTRL-Z, which means pressing the CTRL and Z keys at the same time. This signal has to be followed by (RETURN) to carry out the end-of-file command.

This is a simple way of creating short files of text, such as you might use for batch files, or even short memos. The file is stored as ASCII codes, and can be reproduced by using TYPE or PRINT.

When you do this sort of thing, by the way, you don't hear the disk unit spin for each letter you type, and you may find that nothing at all happens until you use the CTRL-Z (RETURN) signal. This is because output to a disk is always gathered up in the memory until either a set of 512 bytes can be recorded or until the CTRL-Z (RETURN) signal is given.

(Further Information: Amstrad Manual, pages 495, 299.)

■ SECTION 98
Copy to screen

Because CON (an abbreviation of the old name console for a combined keyboard and screen) refers either to the keyboard or the screen, you can use COPY in a form such as:

```
COPY NEWTEXT.TXT CON
```

to copy the contents of a disk file *to* the screen. This form of the command is carrying out virtually the same action as TYPE, however, so there's not the same novelty in it.

It's important to note, however, that not all devices are two-way, like CON. If you attempt, for example, to take data from a printer and send it anywhere, you will simply be greeted by an error message. For example, if you type COPY PRN CON, you will get the message '0 files copied', meaning that nothing has been done.

You can, however, type COPY CON PRN, and whatever you type will be placed to the printer, usually when the CTRL-Z (RETURN) keys are pressed to end the file.

It ought to be possible to use commands like COPY AUX CON or COPY AUX FILENAME, so as to pass signals along a serial line, perhaps from another computer. After many attempts, I could not achieve this, and setting protocols with MODE appeared not to help. The fault could not have been in the hardware, because trying the same exercise using the DOS Plus command PIP FILENAME = AUX: was completely successful.

If you want to use your computer for communications through the serial port, therefore, you may find that you are forced to use DOS Plus rather than MS-DOS. This is puzzling, because other versions of MS-DOS have presented no problems of this type.

You can use device names in this way only with the names for devices that are actually fitted into your computer, so that if, for example, you try to send data out from COM2 then the best you can expect is that nothing will happen. The worst is that the computer will seize up completely, so that no key has any effect, and you will have to switch off and restart. The effects of calling on an input/output (abbreviated to I/O) device that is not physically present can be unpredictable, so the rule is – don't!

The only non-existing device that is permitted is the one called NUL. For example, using COPY FILENAME NUL will send a copy of a file to the NUL device.

Sending data to NUL is not something that you are likely to want to use very often, as its application is in testing a program that sends output to a device like a serial port or a printer. If you want to check that the output can be achieved without any problems, you can redirect it to NUL rather than to AUX or LPT1.

When the output is sent, it will be dealt with immediately, because NUL has no effect, nothing is being sent anywhere. If there is something wrong with the sending action of the program, however, and it is not dealing correctly with the output of data, then the machine may hang up at this point. The reason for using NUL is that you can find this problem a lot faster than if you had to wait 20 minutes for the printer to deal with a long piece of data.

(Further Information: Amstrad Manual, page 299.)

■ SECTION 100
Remote keyboards

There is another command that allows the standard input to come from another source. This is **CTTY**, and it would normally be used in the form CTTY AUX to make the computer take all its input from the serial port.

This could allow the PC to be controlled completely from the keyboard of another machine. Since the serial port could be connected through a modem to the telephone lines, the controlling machine could be at any distance away.

One snag here is that you cannot recover control of the machine by pressing any keys, since only the remote keyboard has any effect. It can return control to you by typing CTTY CON.

The other snag that I found was that no form of input seemed to be acceptable, just like the use of AUX with COPY. It may be that a future issue of the System disk for MS-DOS will feature some way around this problem since, as noted earlier, the corresponding DOS Plus commands were trouble-free.

(Further Information: Amstrad Manual pages 345, 432.)

■ SECTION 101
Redirection with the prompt

There is another form of redirection that does not make use of COPY, and is much more specialised. It is used mainly, but not exclusively, for *program* outputs to disk and *program* inputs from disk files, and it allows some control over how the disk file is used.

It is often particularly useful to be able to redirect the output of a program to a disk file, but the use of COPY is not always sufficient, because all you can specify with COPY is the filename for the disk file, and you can't specify that the material to be copied comes from a program. The alternative method allows you to specify a filename in such a way that either a new file is opened, or if a file of that name already exists, the data is added to the end of the existing file. This is particularly useful, as it allows data to be put into a file in small units and assembled in the file.

This alternative form of output redirection uses the > symbol, which can make the command look confusing on the screen because the screen will normally be showing the A> as the DOS prompt at the time when you type the command. As long as you remember that the A> at the start of a line is the prompt and the > anywhere else is redirection, you should not be confused. The > redirection, used alone, takes the form:

```
DIR > DIRFIL
```

and in this example, which is a very useful one, it allows the directory for a disk to be made into the form of a file called DIRFIL.

This new filename of DIRFIL will be created and can then be copied, displayed with TYPE, printed, and used like any other file. *Note* that you could not have used COPY DIR DIRFIL, because DIR is not a device nor a filename, it is a program, and as such COPY does not recognise it as a source of a file.

(Further Information: Amstrad Manual, page 237.)

■ SECTION 102
Other redirections

You can also use the > redirection to send a directory direct to the printer by using the redirection command in the form:

DIR > LPT1

using the standard abbreviation for the printer.

In many ways, this is more satisfactory than using DIR with CTRL-P, because once the directory has been printed out in this way, the action is finished, and you do not need to remember to use CTRL-P to switch off the printer again.

Any program that gives an output to the screen can have its output redirected in this way. This makes it possible to keep files of directories on a disk, so that one disk can act as a directory to all of your other disks. Other standard programs, like TYPE, can be redirected, but there are usually other ways of achieving the same results.

The only problems that you are likely to encounter are when redirection involves the use of the serial port. I have already noted the difficulties that I encountered with COPY and CTTY, and the same problems were found when trying to use the > form of redirection.

■ SECTION 103
Appending data

Using the > symbol for redirection of data to a file makes a difference to the way that a file is treated. For example, if you used DIR > DIRFIL on a disk that did not contain any file called DIRFIL, then the effect would be exactly the same as before, to create a file called DIRFIL and save in it the content of the directory.

If, however, the file DIRFIL already existed, and you typed DIR B: > DIRFIL, then the directory of drive B would be read and its contents *added* to the end of the existing file DIRFIL.

This make it possible to create a file of steadily increasing size from the use of programs that normally deliver data to the screen. You can use any program for this purpose, so that by the command

TYPE DATAFIL > DIRFIL

you could add the contents of a text file called DATAFIL to the existing file DIRFIL.

(Further Information: Amstrad Manual, page 237.)

■ SECTION 104
Redirecting input

The input of data from a file into a program is another common form of redirection. This type of redirection uses the opposite symbol <.

Some programs start with a series of questions to the user that have to be answered before the program will proceed. A common example is a word-processor program that asks you if you want word-wrap, justification, single spacing and so on. In many cases, the user can press a key that provides a preset default answer to each question, but sometimes you want a set of answers that is not the default set, but which is always the same when you use the program.

By using the < redirection, you can instruct the program to get its answers from a disk file that you have prepared rather than for you to type them each time. The form of the command, which would have to be used as the program was called, would be

SHRDLU < SETUP

in which the word-processor program SHRDLU (don't try to buy it, it exists only in my imagination) takes its inputs from a disk file called SETUP.

The only snag here is that the program must allow you to switch back to keyboard input when you need to, and this is not necessarily provided for.

This redirection action was not available in the early versions of MS-DOS (before Issue 2.0) and so is not used in software that was written before then. As always, you can tell the age of a piece of software by looking to see how many features of the DOS it uses!

(Further Information: Amstrad Manual, page 236.)

SECTION 105
Pipes and filters

MS-DOS was the first operating system for microcomputers to feature **pipes** in a really simple form, and it was also the first to allow the use of **filters**, though these were features of operating systems for larger machines for many years.

A pipe is a method of taking the data output of a program so that it can be used as the input of another *program*, not simply redirected to another device, and a filter is some program that alters data.

The filter is the more important idea to start with, because it's easy to see why pipes are needed when you know what filters do. One filter, for example, might be a program that converts lower-case letters into upper-case letters. Any data sent to this filter would have all of its lower-case letters transformed into upper-case letters.

As it happens, this is not one of the MS-DOS standard filters, but there are three which can be very useful, called **SORT**, **FIND** and **MORE**. All of these are programs that exist on the MS-DOS disk, and can be used only if the disk is in the current drive, or if the filter program has been transferred to the disk that uses it.

The simplest is the MORE filter, which simply cuts text into pages which can be displayed on the screen. If data is sent to the screen through the MORE filter, then one screen full of data will be displayed at a time, with the message '—More—' appearing at the bottom of the screen.

The next page can then be displayed by pressing any key. To send a text file through this filter, you have to specify that the contents of the file will be piped to the filter, and it is taken for granted that the output from the filter will be directed to the screen.

For example, suppose that you have a file called READ.ME which contains updated instructions for a program. To read this in paged form you would need to type

```
TYPE READ.ME | MORE
```

This assumes that the MORE program is on the same disk as READ.ME. If READ.ME is on drive B and the Master disk with MORE is on drive A, then you need to use the command in the form

```
TYPE B:READ.ME | MORE
```

if you are currently using drive A. The alternative is to work from drive

►

■ SECTION 105
Pipes and filters

B and use

TYPE READ.ME | A:MORE

The command word TYPE is needed as a way of commanding that the file be read and sent to the screen, and the pipe symbol | causes the output to be sent through the MORE filter rather than directly. Note that there is no need to specify anything following the name MORE, because the | symbol implies the connection of the filter so that it is placed in the existing stream of data from the file to the screen.

You could, of course, then redirect the output to a file by using the < command, but in this example there would be little point in doing so.

(Further Information: Amstrad Manual, page 237.)

■ SECTION 106
Pipe & filter

The pipe and filter can also be used with any other data that can be modified. If you have a disk whose directory is very large, for example, you can use

```
DIR | MORE
```

to ensure that the directory is displayed one page at a time.

Once again, this assumes that the program MORE is on the same disk that is being used for DIR, otherwise drive letters will need to be used. Piping is achieved by creating a temporary disk file, so that you cannot operate piping if your disks are all write-protected or if they are full.

The file that is created is called %PIPEX.$$$, which is, intentionally, not the sort of snazzy title that you would want for your own files. This ensures that there is little or no chance that the file of that name will exist on the disk when a pipe is created, because the pipe file is deleted immediately following the pipe action, leaving no trace on the disk.

If, however, you interrupted a pipe process, as, for example, by pressing CTRL-Break when a piping action was in progress, it is just possible that you could be left with the pipe file on the disk, since the process had not terminated correctly.

If you interrupt a pipe process, it's a good idea to check the disk directory to see if such a file exists, and to delete it if it is present. It is unusual to find any trace of the temporary program, because the deleting action is fast and efficient. You should remember not to make any use of the %PIPEX.$$$ filename in anything you do.

(Further Information: Amstrad Manual, page 258.)

■ SECTION 107
Sorting data

The other widely used filter of MS-DOS is **SORT**, which will arrange text data in alphabetical order . This can be a very useful filter indeed, because it operates rapidly, and it can make many actions that on other machines require elaborate programs, into very simple commands.

For example, to get on the screen a directory that is alphabetically sorted, you type

DIR | SORT

and press RETURN – that's all! If you had a file called INDEXIT consisting of a list of names in any order, then the list could be sorted into alphabetical order by using TYPE INDEXIT | SORT.

You could, for example, save yourself the cost of a word/page-indexing program by the following steps:

1 **Type the command COPY CON INDEX1 (RETURN).**

2 **Now type your entries, ending each with a (RETURN). They might look like:**

Transistor,1
IC,2
PCB,3
Resistor,4
Capacitor,5
... and so on.

3 **Save this as the file INDEX1 by typing CTRL-Z (RETURN) at the end of the file.**

4 **Now type the command:**

TYPE INDEX1 | SORT < INDEX2 (RETURN).

5 **You now have a sorted index, in true alphabetical order, filed as INDEX2. TYPE the file to see the results.**

(Further Information: Amstrad Manual, page 263.)

■ SECTION 108
Pipe, filter, redirect

The use of the filter and the pipe can be modified by redirection in the usual way, as illustrated above. Suppose, for example, that you wanted to sort your directory and then place the result into a file called ALPHADIR.TXT (the niceties of spelling can vanish when you are allowed only 8-character names, but this one just gets by).

A command such as

```
DIR | SORT > ALPHADIR.TXT
```

will do what is needed here, so that when you TYPE or PRINT the file ALPHADIR.TXT, you will find that the directory is in alphabetical order. The directory has been piped through SORT, and then redirected to the disk file.

Once again, you need to remember that SORT is a program on a disk, and it must be on a disk that is in an accessible drive. This is not sufficiently emphasised in the Manual, and I make no apologies for plugging at it once more.

You could equally easily have directed the alphabetical listing to the printer by using PRN in place of the filename in the example above. Another common requirement is to take data from one file, sort it, and then file under another filename. The example above shows one method, but a neater one is

```
SORT < OLDFIL > NEWFIL
```

with the < symbol indicating redirection of the data into SORT and the > showing redirection of the output to NEWFIL. *Note* that the spaces are important – you will get an error message (File not found) if you omit them.

The pipe symbol | is not needed here, because the filter is not being placed in an existing flow of data, such as from a disk file to the screen. In this case, the data paths are being set up individually, so that the redirection signs are enough, with no need for piping.

The SORT action can be modified by two additions immediately following the word SORT. Using SORT/R will make the sort work in reverse order. This will sort a list of numbers so that the largest number is at the top, and a list of words so that they are in the reverse of alphabetical order.

■ SECTION 108
Pipe, filter, redirect

The other option is to specify in which column of text the character will be found that decides the sorted order. If you use, for example, SORT/+5, the sort will follow the order of the fifth character in each line. This can be useful if the list consists of four places for numbers, a comma, and then a word.

The SORT action can also be used as a program in its own right, not simply as a filter between other actions.

(Further Information: Amstrad Manual, page 263.)

■ SECTION 109
The FIND filter

The third filter that is provided as standard with the MS-DOS disk is **FIND**. This, like the other two, is stored on the disk and must therefore be available when called on. Like SORT, FIND can be used on its own. Unlike SORT, most of the applications for FIND are as a program rather than as a filter.

The action of FIND is to find a specified piece of text. This can be found in a selection of disk files, in text typed at the keyboard, or by filter action in the output from another program. FIND is followed by a phrase, between quotes, that has to be found.

Suppose, for example, that you have a text file called READ.ME, which consists of a set of instructions about a spelling-checker. You want to find in which lines of this file the phrase "key" appears. To do this, type

```
FIND/N "key" B:READ.ME
```

assuming that the disk you want to search through is in drive B and the MS-DOS disk is in drive A.

The result of the action will be to produce a list of lines on the screen, with a number in square brackets (like [13]) at the start of each line. The number is the number of the line in the file, and the line shows how the word is used.

If FIND is used alone, it simply produces a list of lines. Using /N as illustrated gives the line number for each line found, and using /C instead will give just the number of lines in which the word or phrase was found, rather than showing the lines. If you use FIND/N/C, then the /C action prevails, and you will not see any of the lines appear. Another option, /V, allows the display of the lines that do *not* contain the specified word or phrase.

■ SECTION 109
The FIND filter

One use of FIND as a filter is illustrated in the manual, and is very useful. By using DIR | FIND "12-12-86", you can get a list of files created on a particular day. You may see some files with numbers instead of names – these are the temporary files used by FIND, and they will have been deleted by the next time you list the directory.

```
A>
HISTORY   DOC      3382  14-12-85  16:27
REGISTER  DOC       510  14-12-85  16:26
GENICOM   PRT       602  14-12-85   2:41
LASERJET  PRT      1299  14-12-85   2:42
OKI84     PRT       269  14-12-85   2:42
PANASNIC  PRT       494  14-12-85   2:45
SKELETON  PRT      1554  14-12-85   2:38
THINKJET  PRT       321  14-12-85   2:46
```

(Further Information: Amstrad Manual, page 252.)

■ SECTION 110
More on batch files

Batch files were introduced in Part 4 as a method of ensuring that a sequence of commands was carried out. At their simplest, batch files are useful, but their real strength comes when the full set of batch commands can be used.

You can create a batch file by using RPED, but for very short files it's easier and much quicker to use COPY CON filename.BAT. If you make a mistake, just record the file (press CTRL-Z, then RETURN) and start again with the same filename.

For the more advanced user, a batch file can be almost a miniature program in its own right, dealing with a complex set of commands on files that can, of course, include such points as redirection, pipes and filters. The most useful of the more advanced batch techniques, however, involves the use of % parameters.

A **parameter** in this sense means something like the name of a disk file, something that is used in the batch file, but is not the same each time that it is used. For example, you might make a batch file that consisted of the two commands:

```
DEL TEXT.BAK
RENAME TEXT.TXT OLDTEXT.TXT
```

and this would be a perfectly valid batch file for a disk that contained files of these names.

You might, however, have disks on which you had files of other names, but always with the requirement to delete one file and rename another. One way out of the problem is to use names that stand-in for the actual names, so that the commands implied that you should delete file number 1 and rename file number 2 to name number 3.

This is just about the form of a general-purpose batch file command. The % sign is used to mean name number, so that the commands would appear as

```
DEL %1
RENAME %2 %3
```

The significance of the numbers is that they show the order of names that you would have to supply following the name FRUIT.

■ SECTION 110
More on batch files

In the original example, if we imagine that this batch file has been recorded with the filename of FRUIT.BAT, then by typing

```
FRUIT TEXT.BAK TEXT.TXT OLDTEXT.TXT
```

you would ensure that the filename TEXT.BAK is used in the %1 position, the file TEXT.TXT in the %2 position, and the file OLDTEXT.TXT in the %3 position.

(Further Information: Amstrad Manual, page 268.)

SECTION 111
Using parameters

The %1, %2 and %3 in the example are called dummy parameters, and the names that you put into the call for the batch file will be substituted in sequence. This does not mean that a name cannot be used more than once, only that the *sequence* of typing the names determines how they are allocated to the parameters %1, %2 and %3 in this example.

You can also make use of the parameter %0, but not very often. The reason is that %0 is the first filename that you type, which is always the name of the batch file itself, FRUIT in this example. For most purposes, then, you will not make any use of %0.

Another point to note is that when you type a name, such as FRUIT, MS-DOS will search for this among the .EXE and .COM files *first*, and then through the .BAT files. If the name you have used is also the name of a .COM or .EXE file, then your batch file will never run.

Take another example. Suppose that we have a file LONGEAR.BAT which contains the following:

```
COPY %1 %2
DEL %1
TYPE %2 | MORE
```

which uses just two parameters, %1 and %2.

You would call (or *invoke*) this by using

```
LONGEAR A:OLDTEXT.TXT B:NEWONE.TXT
```

and this would result in file OLDTEXT on drive A being copied to the name of NEWONE on drive B, OLDTEXT on drive A deleted, and then NEWONE from drive B displayed on the screen.

Note that the filter programs such as MORE can be used in a batch program in this way, despite the statement in the Manual that piping cannot be used in a batch program.

This illustrates that the parameters %1, %2, and so on, are assigned with filenames (or just disk drive letters if need be) in sequence, but need not be used in sequence, so long as you realise that each parameter will be assigned with a name. You can use only the parameters %0 to %9, and for most purposes this is adequate (though, again, %0 is seldom used because of being allocated to the name of the batch file itself).

SECTION 111
Using parameters

It does not necessarily mean that you can use only ten parameters, only that ten is the maximum that can be used at a time. If you have used ten parameters, and need to be able to enter another set, you can re-assign parameters by using **SHIFT**.

Each time **SHIFT** is used, the parameter %0 becomes available for use again, and the others are shifted by one number. This means that what used to be represented by %0 is now represented by %1, what was represented by %1 is now represented by %2, and so on, with the filename that was represented by %9 now no longer in use. It's most unlikely that you will need to use this facility, which I always think causes more problems than it eliminates.

SECTION 112
New commands

One of the most valuable features of batch files is that each batch file virtually adds a new command to your MS-DOS. As far as your use of MS-DOS is concerned, typing LONGEAR OLDTEXT NEWTEXT is no different in style from typing TYPE OLDTEXT, and its effect is to make a program run.

You can therefore add batch files to your MS-DOS disks so that you can carry out the actions that you need most often with the minimum of typing. One good example is the creation of an index for a book (which was illustrated earlier). As it stood, this required rather a lot of typed commands which you might forget. As a batch file, it's much easier:

```
DEL %1
COPY CON %1
SORT < %1 > TMP.TXT
DEL %1
RENAME TMP.TXT %1
TYPE %1 | MORE
```

You can name this file, for example, as INDEXIT.BAT, and run it by typing its name along with the filename you want to use, such as INDEX.

The action will then be that any existing file called INDEX will be deleted, and then the keyboard can be used to create the new index file. When you end this by typing CTRL-Z (RETURN), the file is saved under the name of INDEX. It is then sorted alphabetically and the sorted version saved as TMP.TXT.

The unsorted file, INDEX, is not deleted, and the sorted file, TMP.TXT is renamed as INDEX. Finally, this file is typed on the screen so that you can see it, using the MORE filter to ensure that only a page at a time is displayed. Even this is clumsier than it need be, because the temporary name could be used more extensively, but this has been intended as a demonstration of the use of a batch file as a command, rather than an example of neatness.

One weakness of this example is that what you see on the screen is not helpful. You see the commands as they are executed, which is not really necessary, but you get no friendly advice, such as 'Now type

►

your index' at the COPY stage. Later in this Part we'll look at how such messages can be delivered.

Nevertheless, this is an advance on the original method, and the file it created could be edited by RPED to check for mistakes, and then used to print an index. You could even include RPED in the batch file rather than use COPY to create the index. It's a good example of how knowledge of MS-DOS can be put to your own particular uses.

■ SECTION 113
AUTOEXEC.BAT files

Any file that carries the name of AUTOEXEC.BAT is a very special file, and there must only ever be one file with this name on a disk. You can, however, have several disks, each with a file called AUTOEXEC.BAT, but with these files entirely different.

The significance of the AUTOEXEC.BAT file is that you never have to type the name AUTOEXEC to run this batch file. It will run automatically on any disk that contains the MS-DOS operating system, immediately after the operating system has been loaded.

Such a batch file can therefore be put on any disk that has been formatted using FORMAT/S, or any disk on which you have used the SYS command to place the MS-DOS operating system into the first few tracks.

Take a look, by way of introduction, at the AUTOEXEC.BAT file on your MS-DOS system disk. It reads:

```
ECHO OFF
KEYBUK
MOUSE
REM POSSIBLE OPTIONS INCLUDE:
REM GRAFTABL
REM GRAPHICS /R
```

and the three commands are carried out each time the Master disk is loaded.

The ECHO OFF is something that we'll come to shortly – it switches off the screen copy of each command in the batch file. The KEYBUK program makes the keyboard of your PC1512 follow the UK pattern, for example, by having the English pound sign. The MOUSE command loads in the MOUSE program, and if, like me, you never use the mouse, you can delete both the MOUSE program and this command from your copy of the MS-DOS disk.

The lines that start with REM (reminder) are ignored when the file runs – they are reminders to you when you read it of other items that can be put in if you want to copy graphics screen displays to your printer.

■ SECTION 114
Off your own BAT

Suppose that you copied the MS-DOS tracks to a new disk, using FORMAT/S, and then copied also a word processor called SHRDLU. Up till now, you would have been resigned to starting up MS-DOS separately, and then loading SHRDLU by typing its name. Now you can proceed differently.

Using COPY CON AUTOEXEC.BAT, you can make yourself an AUTOEXEC.BAT file of this form:

```
ECHO OFF
KEYBUK
SHRDLU
```

which will ensure the use of the UK keyboard, and then run your word processor. You need to put KEYBUK in, because it will not be run otherwise on this disk.

The disk is then used by resetting the machine (ALT, CTRL, DEL keys together, with no disk in the drive), and then inserting this disk when requested to insert a system disk. The system will load, and you will see your word processor SHRDLU up and running.

The usefulness of an AUTOEXEC.BAT file is that it avoids repetitive action. For example, you might always use your computer with a printer that you want to set into bold face. This would be done by running a program, perhaps called PRINBOLD, each time you started work.

By incorporating the line PRINT PRINBOLD into your AUTOEXEC.BAT file, the PRINBOLD program would be run automatically. This assumes that you arranged things so that the printer was switched on along with the computer. If the printer is not switched on at the time when you run a PRINT command, the command cannot be carried out.

This makes it important to be able to include messages and pauses into any type of batch file, something that we'll look at shortly. Note that *any* AUTOEXEC file can be edited using RPED, and you can add to the file on your Master disk. Do *not*, however, make use of any AUTOEXEC file until you are sure that it is correct. A good way of testing is to rename the file as TEST.BAT, and try its effect typing TEST) before using it in the AUTOEXEC.BAT form.

■ SECTION 115
Advanced batch commands

The use of parameter numbers such as %1, %2 and so on, can greatly increase the control that you can achieve along with a batch command, but the story is by no means over. There are several command words that are used in batch files, and can be used *only* in batch files.

These commands are often called batch subcommands, and any attempt to use some of them other than in batch files will cause error messages. If you don't know any programming languages, it is less easy to learn the use of some of these commands, but on the other hand, if you do know a programming language such as BASIC 2, you have to remember that though the batch subcommands are similar to some BASIC 2 commands, they are by no means identical.

We'll start off gently with ECHO, which controls the screen display during the execution of a batch file. The simple straightforward use of ECHO is in the two forms ECHO ON and ECHO OFF. By using ECHO OFF, you will see a lot less appear on the screen while a batch file is executing, because no filenames will be displayed.

This does not, however, turn off all screen display. You would certainly not want to have error messages turned off, for example, and ECHO OFF does not do this. If, for example, your batch file contained the command DEL PIPPLE , and the file PIPPLE did not exist, then you would still see the message 'File not found' when the machine attempted to delete this file. Note, however, that this error does not cause the batch process to stop.

Similarly, other messages that are produced while the DOS commands are being carried out are not hidden. Note that these messages do not stop the action of the batch file. A batch file will therefore normally start with ECHO OFF, and ECHO ON might be used later only if there is some particular need to see what files are being executed.

(Further Information: Amstrad Manual, page 269.)

■ SECTION 116
Talking back

There is one other use of ECHO, which is to display a message that is built into the batch file. Suppose that the batch file at one point starts a lengthy operation, such as comparing disks in two drives to check that they are identical. Whenever a computer starts something like this, it's a good idea to present a message on the screen to warn the operator that something is happening, will take some time, and should not be interrupted by pressing any keys.

To present the message, you put into the batch file just before the start of the comparing action the command ECHO followed by a space and then the message, for example:

```
ECHO Please wait – comparing disks
```

so that this message will appear on the screen and persist while the action is going on.

This type of message *cannot* be suppressed by using ECHO OFF, and if you want it to be cleared off after the compare action is completed, you will need to make the batch command following the compare action a CLS, to clear the screen.

A typical batch file of this type might be named CHECK.BAT and consist of the lines:

```
ECHO OFF
ECHO Please wait...comparing
COMP %1 %2
ECHO All done
```

which would require you to type the name of the batch file, followed by the names of the files to be compared, separated by spaces.

You might, for example, type CHECK A:DIRFIL B:DIRFIL, to check that two files on two different disks were identical. You would see the message ECHO OFF, then the message about waiting, and then the comparison would be carried out. The message from the COPY program would be displayed, because such messages are not suppressed, so that you could tell whether the copies were identical or not. Don't be tempted to put a CLS into this type of program, because it would wipe the message before you had time to read it.

■ SECTION 117
Reminders

There is an alternative to the use of ECHO for messages, in the form of **REM**. A REM command word in a batch file is followed by whatever message you want to deliver on the screen, just like the use of ECHO for messages.

The important difference is that REM messages *are* suppressed if you use ECHO OFF, whereas ECHO TEXT messages are not. The use of REM messages in batch files, along with ECHO OFF, allows you to leave reminders for yourself in the batch file, but prevents these messages from cluttering the screen.

For example, you might like to write a part of a batch file in the form:

```
REM delete all BAK files
DEL *.BAK
REM rename TXT to new extension
RENAME *.TXT *.%1
```

so that when you read the batch file (using TYPE or PRINT), you can see how it works at once without having to go through each command in detail.

The alternative is to write the REM lines in the form:

```
REM Deleting BAK files
(Actions...)
REM Renaming TXT to new extension
(Actions...)
```

with no ECHO OFF at the start of the file, so that the REM lines appear as messages on the screen to remind you of what is going on, and are also present as reminders in the batch file when you TYPE it or PRINT it.

Unless your batch files are very simple, it's important to put in REM lines as a reminder to yourself, because it is remarkably easy to forget exactly what a batch file does after a few months. You will be glad of your REM lines when you find that there is some action that you would like to add to the file, or you want to change the file, or make a new copy as the basis of another batch file.

(Further Information: Amstrad Manual, page 277.)

■ SECTION 118
Repeats

Very often, a batch file must repeat a set of commands until some condition is satisfied. The most common requirement is to carry out an action with each one of a set of different files.

This could, of course, be done on an individual basis, using commands, or on a batch basis by putting the name of each file into the batch command. All of this is a waste of valuable human time, however, and it's much better done by the machine.

The batch command that uses the words **FOR**, **IN** and **DO** is arranged to carry out just this type of repetition action on a group of files. The snag is that the form of the command is by no means simple if you have never programmed, or if you haven't encountered this type of command in programs such as dBASE 2.

The command consists of the words FOR..IN..DO, and the FOR is followed by 'variable name'. This consists of two percent signs followed by a letter, so that %%a, %%b, %%c and so on are all valid variable names. The point of this is that during the execution of the command, this quantity, the variable name, will take the name of each file in turn from a list or set of files that you provide as parameters in the batch file.

By using a variable name such as %%a , you can command actions on that file, such as DELETE %%a, TYPE %%a, PRINT %%a and so on. This means that whatever action you have specified for %%a will be carried out on each file in the set in turn. Note carefully the difference between this and the use of %0, %1, %2 and so on in the command to start a batch file. These are parameters passed into the batch file command by you when you type the name of the batch file and want to specify the file names that it will work on. The %%a, %%b, %%c are *internal* parameters that are used to specify a filename from a list that is contained inside the batch file.

(Further Information: Amstrad Manual, page 271.)

■ SECTION 119
Using FOR

Following FOR %%a, you need to specify the set of files that will be used, and whose names will be represented in sequence by %%a (or whatever letter you have used). This is done by typing IN, and then the set of filenames in brackets.

The set can be specified by using a wildcard, so that you can type IN (*.TXT) for all files using the TXT extension, or you can specify a list of files, with the names separated by a space, such as:

IN(CHAP1.TXT, CHAP5.TXT, CHAP10.TXT)

If you have used a wildcard, each file that corresponds will be represented in turn by %%a and whatever actions you have specified carried out.

When the list is more selective, as in the second example, only these specified files will be used. Finally, you need to specify what has to be done with each file, and this is whatever follows the word DO . If, for example, you used DO DEL %%a , then each file from the list would be deleted in turn, whereas if you used DO TYPE %%a , then each file would be displayed on the screen.

Try, for example, this batch file with the name REPEAT.BAT:

```
ECHO ON

FOR %%a in (*.BAT) DO type %%a
```

This will show all of your .BAT files on screen when you type REPEAT. By using ECHO ON, you can see the title of each file, rather than just a list of instructions.

You need to use CTRL-S to stop this list from scrolling because the filters, such as MORE, cannot be used in this type of command. This is unfortunate, but acceptable, because the use of CTRL-S to stop the listing, and any other key to restart it, is quite easy.

Another restriction is that the command that follows DO has to be a single command. You cannot list a set of commands to be carried out in sequence here, only a single command and any variable names that go with it.

(Further Information: Amstrad Manual, page 271.)

■ SECTION 120
Conditions

By using FOR..IN..DO you can ensure that actions will be repeated for as long as there are files in a list to work on with a single command. This is a simple type of repetitive action that does not allow much in the way of variation.

A repetitive action can be even more useful if there is some method of testing whether or not you want the action carried out subject to some condition. The command words that are used in this way are **IF** and **GOTO**.

The IF word is followed by a test, which can be one of three types, and depending on the result of the test (which will be TRUE or FALSE), then the program can either continue with whatever follows the IF part, or it can GOTO another part of the batch file. This part can skip some commands that follow the IF section, or it can return to an earlier part of the batch file, so that part of the file is repeated until the IF test is satisfied.

The point to which a GOTO refers is marked by a **label name**, a word of up to eight letters that follows a colon. This word will be used in a line of its own to mark where the label refers to, and will also be used following GOTO, but without the colon.

You could, for example, use the word THERE as a label, as illustrated below, in a file called GOTEST:

```
ECHO ON
IF NOT EXIST INDEX1 GOTO THERE
DEL INDEX1
:THERE
RENAME %1 INDEX1
```

and because the echo is left on, you can follow the steps. If you type GOTEST, followed by a space and the name of some earlier text file, you can see what happens. If the file INDEX1 exists, then the step DEL INDEX1 will appear, thanks to the use of ECHO ON. If this file does not exist, then the delete step does not appear. Since the program creates a file called INDEX1, you can run it both with the file not existing, and then with the file existing, to see the difference.

This shows that we use GOTO THERE following the IF test, and :THERE in the line to which the action will shift when the IF test is satisfied.

(Further Information: Amstrad Manual, pages 273 – 5.)

■ SECTION 121
Using IF for files

What you can do with this type of testing depends very much on what can be tested with IF. MS-DOS allows you to test three conditions. The most important, used in the example above, is to find if a file exists.

The test, IF EXIST filename, will be TRUE if the filename is in the same directory, and whatever follows the IF test will be carried out.

To test for a file not being on the disk, you can use IF NOT EXIST filename , which will allow the following action to be carried out if the file is *not* present. This is the form of the test used in the example.

Use of this type of test avoids the error messages that appear when you try to delete or rename a file that is not on the disk. Remember that you can specify a drive letter and path for files that would not appear on the disk in drive A.

You can use the result of the IF test to deliver a message, and wait for you to insert the correct disk – we'll look at that point later. The waiting and the message can be commanded in one operation by using the command **PAUSE**, which is followed by the message.

You might, for example, following a test for the existence of a file as shown above, have the lines:

```
PAUSE Wrong disk – please insert correct disk and press any key
GOTO TRY AGAIN
```

using the label name of :TRYAGAIN to indicate some point in the batch file which comes earlier than the test for the file existing, or possibly just the file test itself.

■ SECTION 122
Error number testing

Another item that can be tested by IF is the error number of a command that has just been executed. This will be 0 if the command has been executed without an error, but some other number, listed below, if there has been an error. Not all commands make use of error numbers. A digit typed during a pause will count as an error number for the purposes of this test.

FORMAT
3 User terminated with CTRL-Break.
4 Serious error (faulty disk).
5 Hard disk formatting abandoned.

REPLACE
1 Error in command line.
2 File(s) not found.
3 Path to a file not found.
5 File write protected.
8 Insufficient memory to operate in
15 Incorrect drive specified.

XCOPY
1 Files not found.
2 User terminated wit CTRL-Break.
4 Specification error (file, path, etc.).
5 Disk error caused abort.

This type of test allows you to test for some specified error, using a line such as

```
IF ERRORLEVEL 2 GOTO mistake
```

This feature is rarely used, because these are actions that you seldom carry out in batch files.

■ SECTION 123
Other tests

You can also check if one piece of text (usually referred to as a **string**) is equal to another, using the double-equals sign (==) as the symbol for comparison. This test can make use of filenames represented by parameters like %1 that you have typed, or filenames taken from a list and represented by variables like %%a .

A more common use is to check the value of an 'environment parameter'. This is a form of name that you use and define for your own purposes. It's a feature more likely to be used by professional programmers rather than in your own batch files. The value of these environment parameters is established by using the SET command.

(Further Information: Amstrad Manual, pages 274, 261.)

PART EIGHT

Programmer's utilities

■ SECTION 124
Utility programs

You can spend many years with MS-DOS without ever using more than a handful of the utilities that are contained on the System disk. In recognition of this, the MS-DOS Master disk that comes with the Amstrad PC omits a lot of the utilities that are intended for the user who programs directly in the codes that operate the machine directly, called **machine code**.

This type of user is in a minority, though certainly not such a small minority as to become an endangered species, and many users of MS-DOS programs will find that they never need these utilities. On the other hand, even if you never use them, it's always useful to know how. I don't often need to make use of a trolley jack for my car, but I certainly would not like to be without it, and I feel better for knowing how to use it.

Programmer's utilities are like this, and some of them can be of very positive benefit if disaster strikes, such as the loss of text discussed earlier. Unlike the programs that you normally work with, however, there's nothing 'user-friendly' about the programmer's utilities – they are intended for the knowledgeable user, and there are no concessions to the beginner.

Using one of these utilities is like reading an article in a trade magazine – if you already know what its all about, you can follow it, but it would be very tough going if you were trying to learn it all from scratch. This Part is certainly not intended to make you an expert on the use of these utilities, but it picks out ones that can be useful and illustrates how they be applied for relatively simple tasks.

■ SECTION 125
The programmer's utilities

Some of the utilities, such as **EXE2BIN**, are so specialised that you are most unlikely to use them unless you are setting out to program in machine code, and all we are looking at here is the one utility, DEBUG, in this class that *can be* of more general use.

Don't think that this is all that is available, it's just that there isn't room on the Master disk for all the utilities of this type that exist. You can, for example, buy disks of utilities, mainly for actions like checking disk contents and recovering lost files for a disk.

The most renowned of these is Norton's Utilities, which has the reputation of being able to recover data from almost any disk unless the system tracks have been corrupted. In the UK, HiSoft also market a disk called 'Knife 86' which offers many useful actions of this type.

You will have to decide for yourself whether your use of the PC1512 will be so intensive that you might need such help, and if you really want to get so involved. If you bought your PC1512 from a dealer who made it clear that there would be absolutely no help available, then these utilities can be a life-line to you.

■ SECTION 126
DEBUG

Of the few programmer's utilities on the System disk, the one that you are most likely to need, is called **DEBUG**. In programmer's language, a bug is a fault in a program, debugging is removing bugs, and the cause of the bug is called, of course, a programmer.

Using DEBUG *definitely* requires some knowledge of the machine and how it is designed, which is why it isn't a utility that you can use like DIR, COPY or DEL. When DEBUG is loaded and run, it loads into the machine, using the lowest part of memory that is available.

Memory in computers is divided into units called bytes, each of which can hold one code number whose value can be anything from 0 to 255. One byte, for example, is sufficient to hold any one ASCII code number, and when a file of text is loaded into the computer, a chunk of memory will then consist of bytes of ASCII code numbers.

If a program is loaded in, then the numbers in the bytes will be in the range of 0 to 255 rather than the restricted range of ASCII codes. So that each byte in the memory can be identified, each one is given a number, called its address number. DEBUG needs to make use of these address numbers for most of its actions.

When you load DEBUG, you can specify also that another file will be loaded with it. For example, you can type DEBUG B:RPED.EXE so as to load in DEBUG, and also load in the program RPED for DEBUG to work on. You would do this only if you were a programmer wanting to check or alter some part of the RPED program, and if you are not a programmer it's more likely that you will not want to load in a file along with DEBUG, or if you do, it will be a text or other ASCII file.

If, as is most likely, you want to use DEBUG only for investigating the contents of the memory of the PC1512, then all you need do is type DEBUG and press the RETURN key in the usual way.

■ SECTION 127
Starting DEBUG

Once DEBUG is up and running, all you see to remind you is a 'prompt' that consists of a hyphen, - . When you see this reminder, you can give a command to DEBUG, consisting of a single letter followed by pressing the RETURN key. Some of the letter commands that can be used, in approximate order of usefulness to the non-programmer, are listed here.

- **d** (Dump memory) Show contents of 128 addresses in sequence.
- **n** (Name) Specifies the name of a file to be loaded or saved. The full name, including extension, must be used.
- **l** (Load) will load the file whose name has been typed in (see n). The filename can also be specified when DEBUG is entered. Another form of Load will load specified sectors directly from the disk
- **e** (Enter) Used to insert or replace numbers in the memory or in a file.
- **s** (Search) Searches for a pattern of numbers or characters

From this list, you will see that the command that is most likely to be useful is the dump command.

SECTION 128
Memory dumps

A memory dump means a listing of each address in memory with the value of the code number that is stored in that byte. Obviously, a listing of the whole memory in this way would be impossibly large, because the PC1512 can use over a million memory addresses, though only 524,288 are used in the standard machine. The dump command therefore displays in pages, consisting of 128 bytes of data at a time.

The big problem is how to find what you are looking for. Looking for some piece of text, for example, in half-a-million bytes sounds rather worse than looking for a needle in a haystack. Fortunately, there are some guidelines. Each address number in the PC1512 is formed by adding two numbers in a rather complicated way.

We can normally ignore this, because DEBUG presents the numbers in the form of two numbers separated by a colon. These numbers are not the familiar numbers that we use in everyday counting, but a number scale called *hexadecimal* (hex) which counts in sixteens. This means that the digits 0 to 9 and the letters A to F are used, and the reason for the use of this scale is that it is particularly well suited to displaying and converting numbers that are stored in the computer.

For most purposes it is the second of these address numbers, the one following the colon, that is important when we use DEBUG. If you have loaded DEBUG along with a file, the second number will be set so that it corresponds to the start of the file. For a file with a COM extension, for example, the second address number will normally be 0100 , which in hex does not mean 100 but 256. If you have loaded DEBUG on its own, the number also starts off at 0100, and to read any data in the memory, you either have to reset it, or use 'd' and keep pressing keys (any key) to display another 128-byte section until you get to whatever you want.

SECTION 128
Memory dumps

```
A>dir b:

 Volume in drive B is #524 V1
 Directory of  B:\

CA
1990:0100  01 00 00 00 00 00 00 00-00 00 00 20 30 31 32 33   ........... 0123
1990:0110  34 35 36 37 38 39 41 42-43 44 45 46 61 62 63 64   456789ABCDEFabcd
1990:0120  65 66 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ef..............
1990:0130  00 00 00 00 00 00 00 00-00 00 00 15 00 55 52 51   .............URQ
1990:0140  53 50 57 56 06 1E 8B EC-0E 07 BF 26 00 8B 6E 16   SPWV.......&..n.
1990:0150  3E 8B 76 00 33 DB E8 FA-01 AC 3C 25 74 30 0A C0   >.v.3.....<%t0..
1990:0160  74 05 E8 BD 01 EB F2 E8-D9 01 1F 07 5E 5F 58 5B   t...........^_X[
1990:0170  59 5A 5D 2E 8F 06 22 00-2E 8F 06 24 00 58 2E FF   YZ]...".....$.X..
-
1990:0580  0E 1F 52 9A 3D 00 A0 19-B8 FF 4C CD 21 EB 2A 4D   ..R.=.....L.!.*M
1990:0590  53 2D 44 4F 53 20 4C 41-42 45 4C 20 50 72 6F 67   S-DOS LABEL Prog
1990:05A0  72 61 6D 20 56 65 72 73-69 6F 6E 20 31 2E 30 32   ram Version 1.02
1990:05B0  20 34 2F 33 30 2F 38 35-00 B4 30 CD 21 86 E0 2E    4/30/85..0.!...
1990:05C0  A3 00 00 3D 0A 03 72 05-3D 14 03 76 0E BA 3D 04   ...=..r.=..v..=.
1990:05D0  0E 1F B4 09 CD 21 06 33-C0 50 CB B4 0D CD 21 BE   .....!.3.P....!.
1990:05E0  80 00 AC 8A C8 32 ED E3-27 AC E8 88 FF 75 05 E2   .....2..'....u..
1990:05F0  F8 EB 1D 90 4E 83 F9 02-72 23 80 7C 01 3A 75 1D   ....N...r#.|.:u.
-
1990:0880  53 54 65 64 20 6F 72 20-41 53 53 49 47 4E 65 64   STed or ASSIGNed
1990:0890  20 64 72 69 76 65 0D 0A-07 00 7C 04 0D 0A 49 6E    drive....|...In
1990:08A0  76 61 6C 69 64 20 64 72-69 76 65 20 73 70 65 63   valid drive spec
1990:08B0  69 66 69 63 61 74 69 6F-6E 0D 0A 07 00 AC 04 0D   ification.......
1990:08C0  0A 07 49 6E 76 61 6C 69-64 20 63 68 61 72 61 63   ..Invalid charac
1990:08D0  74 65 72 73 20 69 6E 20-76 6F 6C 75 6D 65 20 6C   ters in volume l
1990:08E0  61 62 65 6C 00 CF 04 0D-0A 56 6F 6C 75 6D 65 20   abel.....Volume
1990:08F0  6C 61 62 65 6C 20 28 31-31 20 63 68 61 72 61 63   label (11 charac
```

■ SECTION 129
Altering memory

Whether DEBUG is operating on a file that has been left at the bottom end of the memory because of a problem with some other program, or is working on a file that was loaded along with DEBUG, the DEBUG program can normally be placed in a different part of the memory.

This is because when you load DEBUG along with another file, using a command like DEBUG FILPROG.COM, the DEBUG program loads in to another part of memory and then loads in the other program. This means that the other program can be located in the part of memory where it is normally located when it is loaded for running in the normal way.

This is important, because otherwise it would be difficult to be sure of what we were doing. When DEBUG is loaded by itself, with no other filename, it loads into addresses whose second half starts at 0100, as we have seen. Either way, the data that you are working on will be in the memory addresses that it normally occupies.

This allows us to do things like altering the codes in the memory, and saving the altered copy. This is one way of making a *custom* copy of a program for our own purposes, but it is definitely not for the beginner or the user who feels that backing-up disks is a waste of time. You must never attempt any of these actions on any file or disk for which you do not have an adequate backup – and I usually keep two backup copies of anything that I want to alter in this way. Apart from the risk of losing a diskfull of data, there's nothing to prevent you from experimenting.

It can't damage the computer, and if you lose a disk for which you have a couple of backups, then all you have really lost is time. On the other hand, you may very well have achieved something very useful.

■ SECTION 130
Printer control

(Advanced work; omit on first reading.)

As an illustration, try the following. The aim is to produce a file that when printed using PRINT or TYPE (with CTRL-P), will set an Epson printer into bold face.

This is done by sending the ASCII codes 27 and 69 (code for letter E) to the printer, in that sequence. At first sight this looks very simple, and all you have to do is create a file with these characters.

The problem is that very few file editors will allow you to enter the ASCII Esc code which is 27. The solution is to enter some dummy value, and then use DEBUG to change the file. Start by calling up RPED, specify a new file and name it BOLPRT, then make a file that consists only of a space and the letter E, whose ASCII code is 69.

Now proceed as follows:

1 **Enter DEBUG, using your filename. For example, use DEBUG B:BOLPRN if the printer file is on drive B.**

2 **Type d (RETURN) to see the file displayed. It should read:**

20 45 0D 0A 1A 20 ...

with the rest of the file giving the 20-character code, which is the space.

3 **The character 45 is E in ASCII code. This is 45 and not 69, because DEBUG uses hexadecimal codes.**

4 **Type e100 (RETURN). You will now see the number 20 displayed with a dot following it.**

5 **Type 1B and press the spacebar. This has the effect of changing the 20 number to 1B, the Esc code. Now press RETURN.**

6 **Press d100 (RETURN) to check that the change has been made.**

7 **Type w (RETURN). This will return the file to the disk in its altered form.**

8 **Type q (RETURN). This gets you out of DEBUG and back to the normal DOS prompt.**

■ SECTION 131
Testing the file

(Advanced work; omit on first reading.)

If you have an Epson printer on line, you can now test the BOLPRT file. Switch the printer on, and select a short file, such as GEM.BAT. Type TYPE GEM.BAT, press CTRL-P, and then (RETURN). The printer should print out the file in ordinary type. Press CTRL-P again.

Now type TYPE BOLPRT, press CTRL-P, and then (RETURN). The printer will take a new line, but probably won't print anything. Press CTRL-P again, and try printing the GEM.BAT file again. This time it should be in bold type, indicating that your BOLPRT file did as it was supposed to.

Your printer manual will show details of all the codes that alter printer behaviour. Many of these will start with the Esc character, which is 1B in hexadecimal code, so that you can write a string of codes in this way to set up whatever you want. The general method that has been illustrated here will work just as well for any sequence of codes.

You create a file with RPED, and use a space for any character that you cannot enter, such as Esc. By using DEBUG with this file, you can then substitute these characters, and save the amended file. This file can then be printed as part of an AUTOEXEC.BAT sequence if you want your printer always set in this way.

■ SECTION 132
Disk recovery

One particularly useful feature of DEBUG allows sectors of data stored on a disk to be loaded into the memory of the computer. This is one way in which data from a disk that has been damaged can be recovered, and it is the basis of the way in which a deleted file can be replaced.

In this book, which is not intended for experts, we cannot deal with the problems of recovering a deleted program file, because such problems do need considerable knowledge of the system, but we can certainly look in outline at how to recover deleted text files (ASCII files).

The trouble with this command, a variety of the LOAD command, is that it is never easy to work with, certainly not so easy as the disk utility programs that are sold specifically for the purpose. What follows, then, is a *very* brief description which is intended to be the basis for experiment rather than a do-it-yourself guide to any kind of file recovery from a damaged disk or deleted file.

To start with, the sectors on the disk are numbered starting with 0 and going up to 719. In hex, this latter number is 02CF. The number is not quite so straightforward as you might think, however, because of the way that the double-sided disks are used.

Sector 0 is on the first side of the disk, and is the first sector of the first track (see Part 1 if you have forgotten about tracks and sectors). There are 9 sectors on each track, so that the numbering of sectors goes from 0 through 1, 2, 3...up to sector 8, which is the last sector (since we started at 0) on Track 0, first side. The count then continues, but sector 9 is the first sector of Track 0 *on the other side* of the disk, and in this track we find sectors 9 to 17. Sectors 18 to 26 are on Track 1 on the first side, and sectors 27 to 35 on Track 1 of the second side and so on.

The loading direct from sectors depends on making use of these numbers, but in the hex scale. The best way to experiment is on a disk that has been formatted with no System tracks, and has been used only for text.

■ SECTION 133
Reading errors

(Advanced work; omit on first reading.)

The l (small L) command of DEBUG, as used for loading disk sectors, requires four numbers following it, all in hex. The first number is a starting number in memory, and a good choice is the usual 100. The next number is 0 for drive A, 1 for drive B, 3 for drive C or 4 for drive D. The third number is a starting sector number. For a disk of text, a good choice is C, because this is sector 12, the first one that is likely to be used. Sectors up to this are kept clear in case you want to add the system tracks. The fourth number is the number of sectors that you want to read. You can try 10 which is sixteen sectors (10 hex is 16 in normal numbering).

With DEBUG running, type:

L 100 0 C 10 (RETURN)

and the disk will spin briefly, leaving you with the DEBUG prompt again.

Now type:

D 100 (RETURN)

and you will see the start of the text on the disk.

You can now alter the text, using E as in the example earlier. For a text disk, this is hardly worthwhile, because you could do the same with less hassle by using RPED.

The point, however, is to show that DEBUG can deal with anything, and that includes items that text editors cannot deal with, like the Esc character, and files that are on a disk and which might be corrupted. The file can be put back on the disk in its altered form by using the W command.

If, for example, you use DEBUG to replace one character in a file with the number 1A, which is the end-of-file marker, the file can from then on only be read as far as this point. If a file cannot be read correctly by a text editor, it may be that some disk corruption has placed the 1A character on the disk, and that you could find it and remove it.

In general, though, the use of DEBUG in this way is for the seasoned programmer. The main reason for including this description is to show how much more remains to learn after you have taken off your L-plates.

APPENDICES

■ APPENDIX A
Assorted commands

In addition to the main commands of MS-DOS that we have covered in this book, there are some that are *odd* in the sense that you seldom need them, and they don't fall into any particular group. In this Appendix we'll look very briefly at some of these commands.

The **BREAK ON** command is a way of getting better service from your use of CTRL-Break. When a program makes extensive use of disks, the action of testing for these keys being pressed is suspended during disk operations. If you have typed and entered BREAK ON, the testing rate is increased and extended, making it easier for you to stop a program.

This also slows down the program action, and you can return to normal service by typing **BREAK OFF** (RETURN).

You can set up a different prompt, if the sight of A> does not appeal to you. This takes a lot of typing, but could be done in a batch file, as it must be repeated each time the machine is switched on or reset. The command that is used is **PROMPT**, and a full example is included in the Manual, page 352.

The command **SET** is used to 'set an environment parameter'. This means that a word can carry a code that can be used in a program or a batch file. It's most unlikely that you'll want to use this unless you intend to write programs or make use of unusually elaborate batch files. The command is dealt with in the Manual, page 261. Whatever you do, don't use the words PATH or COMSPEC in a SET command of your own unless you know what you are doing.

The **VER** command prints the version of MS-DOS that you are using. For my machine, this was 3.20. If your MS-DOS disk gives any other number, then you may find some small changes in the way that commands work, or that some restrictions have been removed.

The **VERIFY** command is used in the forms VERIFY ON, VERIFY OFF and VERIFY. After you have typed VERIFY ON, any file that is saved will also be checked with the original copy in memory or in another file. This can take some time, so that VERIFY is usually turned off. You can check by typing VERIFY (RETURN).

APPENDIX B
Altering the RAM-disk

The size of the RAM-disk, which is drive C: for machines with no hard disk, is normally set at 34K. This looks rather small compared with the 100-odd K that you get with the smaller PCW 8256 machines, and the reason for using such a small capacity is that the size is limited so that GEM can be used.

If you are by now a confirmed MS-Dosser, you can allocate an enlarged RAM-disk each time the machine is switched on. This size is in a permanent part of the memory, a part that is maintained by the batteries when the machine is switched off. Unlike the use of a batch file, then, any alteration that you make here will affect all the programs that you use.

Altering the command settings in permanent memory involves the use of the **NVR.EXE** program. You should copy this program over from Disk 3 – you may have to delete something less useful from your MS-DOS disk in order to create space. You can delete EXE2BIN and EDLIN without too many qualms.

When NVR has been copied over to your MS-DOS disk, start it by typing NVR. The display on the screen then allows you to choose various options, all detailed in the Manual on pages 476 – 481. You can change the RAM-disk to 128K by selecting this option, typing the number and pressing RETURN.

The RAM-disk new size will not change when you leave NVR, only when you next start up the machine, or reset with CTRL ALT DEL in the usual way.

Note you cannot run BASIC 2, or other programs using GEM, if the RAM-disk size is set to 128K.

Index

Index

Index

Index

STEP BY STEP

Using MS-DOS on the Amstrad PC

Ian Sinclair

For the first time user, new to computers, the process of learning how to use the machine can be a daunting process.

Step by Step books have been designed for clarity and ease of use. They guide the reader gently through the stages of understanding how to use the hardware and software that makes up the Amstrad PC1512/1640 computer system, quickly and effectively.

This book, **Using MS-DOS on the Amstrad PC,** introduces the reader to MS-DOS methods for those users who have some small prior knowledge of computing but no previous experience of MS-DOS or CP/M. The book is divided into parts, each of which deals with a number of topics, one section to each topic. In each section, there is text and examples which show in detail how the commands of MS-DOS are used. The order of the sections follow the logical path of the user who will experiment first with simple operations and eventually work up to advanced uses of MS-DOS. A very full index makes it easy to obtain information.

- The disk system
- Simple internal commands
- Using the MODE command
- Inputs and outputs
- Formatting disks
- Batch commands
- Errors and error messages
- Utilities

Ian Sinclair is one of the most prolific writers of technical and computer books in the world, with over seventy titles to his credit.

HEINEMANN
NEW·TECH

£9.95 net

ISBN 0-434-91842-3

9 780434 918423